

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Mathematical Statistics

Lidija Kirjackaja

RECOMMENDING NEWS USING GRAPH-BASED ALGORITHMS

Master's Thesis

Applied Statistics study program, state code 621G31001
Data Science specialization

Vilnius, 2018

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Mathematical Statistics

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Applied Statistics** study programme Master Graduation Thesis

Title	**Recommending News using Graph-based Algorithms**
Author	**Lidija Kirjackaja**
Academic supervisor	**Dmitrij Melichov**

Thesis language: English

Annotation

The problem addressed in this paper concerns personalized news recommendation based on short user activity sessions. The graph-based algorithms were chosen to be investigated, as they show a growing potential facing the pressing need of modern recommender systems to provide users with explanations of the built recommendations. The aim of the work was to analyze existing algorithms and find the ways to improve their performance in terms of various quality assessment factors.

Well-known graph-based algorithms, such as Random Walk with Restart, PathSim, SimRank, and their variations were implemented, evaluated and compared using different evaluation settings. It appeared to be challenging to beat outstanding recommendation accuracy of a simple popularity-based approach, however, at the end a novel method was proposed which looks promising with regard to both recommendation accuracy and explainability. The proposed way of explaining recommendations is based on graph meta paths and has a clear semantic reasoning behind.

The thesis consists of: 65 p. of text without appendices, 21 figures, 27 tables.

Keywords: recommender systems, recommendation algorithms, graph-based algorithms, news recommendation, explained recommendations

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Matematinės statistikos katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Taikomosios statistikos** programos magistro baigiamasis darbas

Pavadinimas	**Naujienų rekomendavimas taikant grafais grįstus algoritmus**
Autorius	**Lidija Kirjackaja**
Vadovas	**Dmitrij Melichov**

Kalba: anglų

Anotacija

Šiame baigiamajame magistro darbe nagrinėjama personalizuotų naujienų rekomendavimo problema remiantis trumpomis vartotojų aktyvumo sesijomis. Grafais grįsti algoritmai buvo pasirinkti nagrinėjimui, kadangi jie rodo augantį potencialą sprendžiant moderniųjų rekomendacinių sistemų būtinybę teikti vartotojams rekomendacijų paaiškinimus natūralia kalba. Darbo tikslas buvo išanalizuoti egzistuojančius algoritmus ir rasti jų tobulintinas vietas, bei pasiūlyti naujus metodus, pagerinančius rekomendacijų kokybę įvairių kokybės vertinimo kriterijų atžvilgiu.

Darbe buvo įgyvendinti, įvertinti ir palyginti gerai žinomi grafais grįsti rekomendaciniai algoritmai, tokie kaip Random Walk with Restart, PathSim, SimRank, ir jų variacijos. Buvo pasiūlytas naujas metodas, kuris parodė išskirtinius rezultatus ir rekomendacijų tikslumo, ir paaiškinamumo atžvilgiu. Pasiūlytas rekomendacijų paaiškinimo būdas yra grįstas grafo meta-kelių ir yra paremtas aiškia semantika.

Darbo apimtis: 65 p. aiškinamojo rašto be priedų, 21 illiustacija, 27 lentelės.

Prasminiai žodžiai: rekomendacinės sistemos, rekomendaciniai algoritmai, grafais grįsti algoritmai, naujienų rekomendavimas, paaiškinamos rekomendacijos

Contents

1 Acknowledgments **5**

2 Introduction **6**

2.1 Research relevance 6

2.2 Thesis objective and tasks 6

2.3 Outline 7

3 Recommender Systems **8**

3.1 Definition 8

3.2 Classification 8

3.2.1 Content-based filtering 9

3.2.2 Collaborative filtering 9

3.3 Nature of data 10

3.4 Recommendation problem abstractions 10

3.4.1 User-item matrix completion 10

3.4.2 Next-click prediction 11

3.4.3 Recommendation problem abstractions comparison 12

3.5 Sequence-Aware Recommender Systems 13

3.5.1 Benefits 13

3.5.2 Categorization based on context adaptation type 14

3.6 News Application Domain 14

4 Graph-based similarity measures **17**

4.1 Heterogeneous Information Network 17

4.2 Network Schema 18

4.3 SimRank 19

4.4 Personalized PageRank/Random Walk with Restart 22

4.5 PathSim 24

4.6 PathCount 26

5 Building recommendations **27**

5.1 Weighting session articles 27

5.2 Combining short-, medium- and long-term interests 29

5.3 Session-kNN 31

6 Explainable recommendations **32**
6.1 Explanation styles . . . 32
6.2 Hybrid explanations . . . 33
6.2.1 Hybrid recommendations ranking . . . 34
6.3 Item-based Explained Recommendations . . . 36
6.4 Session-based Explained Recommendations . . . 38
6.5 User-based Explained Recommendations . . . 38
6.6 Explainability coverage . . . 39

7 Experimental evaluation **40**
7.1 Evaluation metrics . . . 40
7.1.1 Accuracy measures . . . 40
7.1.1.1 Precision . . . 40
7.1.1.2 Normalized discounted cumulative gain (nDCG) . . . 41
7.1.2 Beyond accuracy . . . 41
7.1.2.1 Intra-list diversity (ILD) . . . 42
7.2 Prequential evaluation protocol . . . 43
7.3 Dataset description . . . 45
7.3.1 General statistics . . . 45
7.3.2 News categories . . . 46
7.3.3 News locations . . . 47
7.4 Popularity-based baseline . . . 49
7.5 Sensitivity Analysis of RWR . . . 50
7.5.1 Changing the time window size w . . . 50
7.5.2 Changing the network structure G . . . 51
7.5.3 Changing the weighting strategy of articles in a session . . . 52
7.5.4 Combining short-, medium- and long-term user interests . . . 54
7.6 Sensitivity analysis of PathSim and PathCount . . . 55
7.6.1 Changing the time window size w . . . 55
7.6.2 Changing the meta path . . . 55
7.6.3 Changing the weighting strategy of articles in a session . . . 56
7.7 Sensitivity analysis of Session-kNN . . . 57
7.7.1 Changing the similarity search model . . . 57
7.7.2 Changing the number of nearest sessions k . . . 58
7.8 Methods comparison . . . 58
7.9 Experiments on explainable recommendations . . . 60

8 Results and Conclusions **61**

List of Figures

1 Distribution of research works per application domain adopted from [20] 14
2 Online News portal Network Schema 18
3 Online News portal Network Schema (detailed) 18
4 Toy example of a heterogeneous graph network USAC 19
5 Possible Meta Paths of the USAC Network starting from A 25
6 Bipartite session-user graph toy example 26
7 Short- and medium- term aware recommendation schema 31
8 Explanation styles for three recommendation strategies 32
9 SAC graph toy example for calculating average number of similar nodes in the graph 34
10 ASA-based Recommendations 37
11 Hybrid 4-D meta path Explanation for IB Recommendation strategy with IB_{All} approach. 37
12 Hybrid 3-D meta path Explanation for SB Recommendation strategy. . . 38
13 Hybrid 3-D meta path Explanation for UB Recommendation strategy. . . 39
14 Prequential evaluation protocol schema 43
15 Distribution of number of articles per session for Video33 dataset 46
16 News category preferences by users of two news channels 46
17 News category preferences dynamics at Video33 dataset 47
18 News category preferences dynamics at Video33 dataset in Val-Venosta/Vinschgau geographic location 47
19 Category preferences in Bolzano/Bozen and Val-Venosta/Vinschgau in the period of April-June 2016 48
20 Comparison of combinations of short-, medium- and long-term interests . 54
21 Models precision comparison on Video33 dataset 59

List of Tables

1 Recommendation problem abstractions comparison 12
2 SimRank similarity statements for different recommendation cases 19
3 Recommendation cases corresponding sub-graphs of the USAC network . 20
4 SimRank similarities of article A1 with other articles on different sub-graphs 20
5 Random Walk with Restart result transition matrix 23
6 PathSim similarities of article A3 and other articles using different meta-paths 25
7 RWR, PathSim, PathCount measures for article A1 with A2 and A3 . . . 26
8 Item relevance vectors for items $a1$ and $a2$ 29
9 Comparison of weight assignment strategies 29
10 Short/Medium/Long-term based model settings 30
11 Meta-paths used in three recommendation strategies 33
12 Number of votes for each candidate article to be recommended based on the number of times that it appears in the meta path connections. 35
13 Candidate Item Relevance 36
14 General statistics of datasets 45
15 Popularity-based algorithm with different t_S 49
16 Average number of articles and sessions per different size of training set . 50
17 RWR evaluation by using different window sizes w 51
18 RWR evaluation by using different network structures G 52
19 RWR evaluation by using different article weight assignment strategies W 53
20 Comparison of S, SM, SL and SML approaches 54
21 PathSim and PathCount evaluation by using different window sizes w . . 55
22 PathSim and PathCount evaluation by using different meta paths 56
23 PathSim and PathCount algorithms by using different article weight assignment strategies W 56
24 Session-kNN evaluation by using different similarity search models 57
25 Session-kNN precision evaluation by using different k 58
26 Models comparison 59
27 Explainable recommendations precision and explainability coverage vs. different explanation styles and recommendation strategies 60

1 Acknowledgments

I would like to express my gratitude to my thesis supervisor Dmitrij Melichov, who supported and encouraged me over two years of my master studies. He helped me with professional advice and made valuable remarks on the work I was doing.

This thesis would not have happened in it's current form without active participation of my external adviser Panagiotis Symeonidis, ass. professor of Free University of Bozen-Bolzano. During the last two semesters his practical advice, professionalism, and outstanding knowledge of the topic gained from over 10 years of experience in recommender systems, made me always stay interested, motivated to learn, and excited to develop something new and valuable to the scientific community.

Finally, I would like to thank the International Department of Vilnius Gediminas Technical University for making it possible to participate in the exchange program and gain a valuable experience of working in scientific environment of a foreign university.

2 Introduction

2.1 Research relevance

Recommendation engines nowadays are an essential part of many online systems working on different application domains, such as movie recommendations, e-commerce, media streaming, etc. Receiving automated, usually personalized recommendations on the Web has become an integral part of our daily online routine.

Most well-known recommendation algorithms consider user-item interactions as the main mean for collaborative filtering, finding similar users and similar items, as well as modelling general user preferences. Only in past few years the interaction between users and items is considered on a session level, providing possibility to model short-term user preferences and short-term item-item interaction. Investigating the use of short user sessions in the news domain for building personalized next-article recommendations is the main focus of this master thesis. We use a combination of graph-based recommendation techniques with sliding window models to find the most relevant news articles to the user, that would conform with her short-term reading goals and general preferences. Moreover, we believe, that making use of graph-based models, in particular, semantic network meta paths, should have a great impact on better explainability of recommended items, which strongly relates to the new regulation of European Union that becomes enforceable on 25 May 2018 and explicitly defines the users' "right to explanation" of algorithmic decisions made about them.

2.2 Thesis objective and tasks

The main objective of this master's thesis is to *develop a framework for providing accurate and explainable personalized news recommendations, making use of graph-based algorithms applied on heterogeneous information networks.*

The tasks to be performed in order to achieve the stated goal are summarized as follows:

- Study the recommender systems related literature, paying the most attention to the latest research works that consider the problems of next-item recommendation, news recommendation, graph-based and meta path-based models in recommender systems, and explainability-concerned recommendation methods;
- Infer the benefits and drawbacks of known graph-based recommendation algorithms, implement them, and test experimentally on various toy examples as well as on a real dataset;

- Clarify the effect on quality of recommendations when changing:
  (i) the size of the training data;
  (ii) the number of participating graph entities;
  (iii) the way of capturing short user interest;
  (iv) other parameters if applicable for the tested method.
- Investigate the means for explaining recommendations using network meta paths, and explore how better explainability affects model accuracy.

2.3 Outline

Chapter 3 of this master's thesis is dedicated to the general analysis of recommender systems, their most common categorization into content-based and collaborative filtering, nature of used data. We focus on next item recommendation problem and sequence-aware recommender systems that are used to solve it. We finish the chapter describing characteristics of news recommendation domain and challenges to be faced.

In Chapter 4 we familiarize with heterogeneous information networks, that can naturally capture both collaborative filtering and content-based approaches in one structure. We then discuss well-known graph-based similarity measures, such as SimRank, Personalized PageRank and PathSim, and propose our variation denoted as PathCount.

In Chapter 5 we describe the ways of building recommendations using the previously defined similarity measures. We first talk about the ways to capture short-term user interest inferred from his current session. Then we propose a method to capture also medium and long-term interests. Finally, we describe a Session-based kNN method, recently proposed in the research field as an accurate next user click predictor.

In Chapter 6 we propose a way to produce explainable recommendations making use of graph meta paths. We describe four explanation styles and three recommendation strategies that can use different explanation styles separately or in combinations. Finally we propose an explainability coverage metric for measuring how explainable are the recommended items to the user session and to the user.

In Chapter 7 we first define the experimental evaluation protocol and used evaluation metrics. Then, we describe the datasets and perform their log analysis. Finally, we perform experiments, assessing the quality of the models with different model settings. We also perform experiments assessing accuracy of our proposed explained recommendation strategies.

Finally, in Chapter 8 we summarize the results of the research work done and make conclusions.

3 Recommender Systems

In this chapter we define the concept of a recommender system, present the difference between content-based and collaborative filtering based systems, distinguish between different types of used data, discuss two recommendation problem abstractions, i.e. user-item matrix completion vs. next user click prediction. Then, we present the sequence-aware recommender systems, used when dealing with the next item recommendation problem, which is the problem to be studied further within this research work. At the end, we analyze the news application domain and challenges to be faced when personalizing news articles recommendations.

3.1 Definition

> *Recommender Systems* (Recommendation Systems, RS) are software tools and techniques providing suggestions for items to be of use to a user [21].

Recommender System is one of the most important parts of any e-commerce ecosystem. While amounts of available information on the Web continue to grow, valuable recommendations become vital for individual users as well as for businesses. Many websites invest a lot of time and resources to provide the best personalized recommendations to their users, to result in general user satisfaction and engagement. For nearly two decades researchers worked on the means to predict people attitudes towards unfamiliar objects in the most accurate way, resulting in a mixed set of algorithms and rich collection of tools for estimating their efficiency.

Internally, such systems analyze the past behavior of individual users or of a user community as a whole to detect patterns in the data [20].

While different domain businesses operate with different objects (e.g. movies, songs, products, articles), we further refer to the term *item* as a general term used to denote what the system recommends to users.

3.2 Classification

Traditional RS classification distinguishes two general approaches used in recommender systems:

1. Content-based filtering (CB);
2. Collaborative filtering (CF).

In practice, hybrid approaches are most often used, taking advantages from both strategies.

3.2.1 Content-based filtering

Content-based (CB) RS build models on the basis of target user historical profile, recommending items that are most similar to the ones he has interacted with in the past (or is interacting right now). Similarity between items in the content-based context usually describes belonging of items to the same or similar categories, or having common descriptive features.

The main advantage of CB is that it alleviates the "cold start" problem [25] - new incoming items can be easily associated with the other items having similar features. For example, if a user several times bought a book of the same author, when the new book of that author is published, it would be automatically recommended to the user.

Disadvantage of using content-based systems alone is a small provided novelty for a user, i.e. weakly diversified recommendations. It becomes hard for users to investigate something new and unseen, as they are always being offered items of similar content, not conforming to the natural human will to change.

3.2.2 Collaborative filtering

Collaborative filtering (CF) techniques are classified in two primary areas: the neighborhood methods and latent factor models. Neighborhood methods are based on computing the relationships between all items and between all users. Latent factor models use matrix factorization techniques to explain user preferences and item features by the means of certain amount of factors inferred from user behavior patterns.

Two types of former CF approach are distinguished: *user-based CF* and *item-based CF*. An intuition behind user-based CF is that people that agree on one issue are more likely to agree also on another. We usually ask for advice from those people, whose tastes usually coincide with our own tastes. Recommendation systems based on CF automatically search for users with similar interests through the whole network and exchange their preferences between them. In the item-based CF approach, given a target user and his historical item interactions, the algorithm searches for the most similar items and recommends the closest ones [22].

The advantage of CF algorithms is that they are very versatile, i.e. can be applied to any domain, without domain modeling based on item metadata. Moreover, most of the times CF algorithms result in better performance in terms of recommendations accuracy, i.e. provide higher personalization.

One of the main disadvantages of CF against CB is the "cold start" problem, i.e. no recommendations can be made for new users, and new items can barely be recommended.

3.3 Nature of data

Datasets, used for learning of recommendation models, store historical interaction records of users with items. Depending on the domain and type of a system, those can be

- explicitly given user feedback (e.g., ratings, likes);
- implicit feedback extracted from web server logs (e.g., clicks, views, purchases).

While explicit feedback seems to be a more reliable source of information and most of the traditional recommendation techniques are developed to predict user ratings for unseen items, in practice, it is a challenging task to make users rate the watched/bought/read items, which results in the lack of information and unreliable predictions at the end.

Implicit feedback, on its turn, shows real users' interests - what kind of products they view, which articles they read, etc. However, it is more difficult to operate with such data, as sometimes clicks might be unintended, influenced by recommendations themselves, or to appear to be out of user interest after he analyzed item content. In this research work we further focus on the algorithms dealing with implicitly given user feedback.

A number of research works exists investigating recommendation techniques for implicit user feedback, however, the problem formulation is often based on user-item preference matrix completion, rather than on sequence-awareness and next-click prediction, which we discuss in detail in the next section.

3.4 Recommendation problem abstractions

Depending on business domain, it's specific and needs, two different general recommendation problem abstractions can be stated:

1. User-item matrix completion
2. Next user click prediction

We further discuss both problem abstractions, although the most attention of this research work is further dedicated to the next-click prediction problem.

3.4.1 User-item matrix completion

Historically, academic research is often based on the *user-item matrix completion* problem, where knowing past user feedback for a number of items, the goal is to predict his opinion about other, new items. A user is then provided with a ranked list of items, where

the rank indicates how relevant or attractive to the user an item is predicted to be. User-item interactions are always assigned to known users and items, and the matrix of known interactions is called *user-item adjacency matrix*. This approach has been researched over several decades, and is widely applied in domains like movie and music recommendations. The abstraction is generally well-suited to train machine learning models that aim to capture longer-term user preference profiles and generate recommendations that best suit general user interests. Evaluation of such recommendations is called *hold-out evaluation*, where some amount of known user feedback is hidden from the model, and is then used to evaluate model accuracy. This is only applicable in offline experiments, and has some drawbacks, such as using "future" user feedback to predict the feedback from the "past".

3.4.2 Next-click prediction

Matrix-completion corresponding algorithms typically do not implement any specific means to take the users' short-term behavior or intents into account. They also make no value of sequentially-ordered user interaction logs, while treating each user-item interaction as a separate unit. To overcome aforementioned problems, and as more and more applications started storing users' *clickstream data*, a new strategy was proposed, designated as *next-click prediction* (or next-item recommendation). Common application domains, where recommendation problem is solved by predicting the next click of the user, are e-commerce online shops, news websites, music portals (next song recommendation). The challenge, and the key to success for such applications is identifying short-term user interests as well as longer-term sequential patterns [20]. A new *session* layer appears in-between user-item interactions.

> *Session* - is a sequence of clicks of a single user within a short enough time interval. It can be a series of product views until the purchase is done, or a sequence of user clicks until he leaves the website.

A typical example problem setting is that of *session-based recommendations* [10, 8, 4] (2015, 2016, 2017), where no longer-term user histories are available (no registered users), or users are not keen on authenticating themselves (e.g. on online news portals), and a system is focusing on single sessions rather than on complete user profiles. The recommendations are adapted according to the assumed short-term interests of an anonymous user. The goal in such scenarios usually is to recommend objects that match a given sequence of current (recent) user actions. Nevertheless, considering such sequences can also be useful in domains where users are known, to derive their longer-term behavior patterns, detect interest drifts, identify short-term tendencies in different demographic user groups.

Obviously, both long-term user profile and short-term preferences are important for better personalization of built recommendations. Resolving a preference integration problem is a challenging task and attracts more and more researchers' attention. For example, Epure et al. in the paper "Recommending Personalized News in Short User Sessions" [4] considers three levels of reading interests based on time dimension: short-, medium-, and long-term, and shows, that recommendations based on combined short-term and long-term interests result in better prediction accuracy, while combination of short-term and medium-term interests yields higher news variety.

The protocol for evaluating next-item recommendations was first proposed in 2014 by J. Vinagre et al. [26] and is known as *prequential evaluation protocol*, designed to evaluate algorithms in streaming environments, suitable also for offline evaluation. The protocol exclusively evaluates how well the model predicts the next observation, ignoring all subsequent ones. Later in Section 7.2 we describe the prequential evaluation protocol in detail, and adapt it to better suit our performed experiments.

3.4.3 Recommendation problem abstractions comparison

Table 1 summarizes main properties and differences between two previously discussed recommendation problem abstractions. Further in this research work we stick solely to the next-click prediction problem.

	User-item matrix completion	**Next-click prediction**
Inputs	User-item adjacency matrix	Clickstream data
Outpus	General ranked list of items predicted to be of interest to a user	A ranked list of predictions for the next user click
Evaluation	Hold-out evaluation	Prequential evaluation

Table 1: Recommendation problem abstractions comparison

3.5 Sequence-Aware Recommender Systems

Under the following subsection we discuss *Sequence-aware recommender systems* (SARS), that are usually used when dealing with next item recommendation problem.

Computational tasks of sequence-aware recommender systems are identification of *sequential patterns* in the recorded user actions (order of actions within the session is considered), or *co-occurece patterns* (important is that two actions happened together within the same session). These patterns, or more generally - learned models, have to be assigned to the time interval on which they are learned, and be relevant at the point in time at which recommendation is made.

3.5.1 Benefits

Two of the benefits, achieved by using sequence-aware recommendations are *context adaptation* and *trend detection*.

Context adaptation. In many domains, the relevance of a recommendable item not only depends on the users' general preferences, but also on their current situation and their short-term interests. Contextual factors can be *representational* (observable) - user's geographical position, the current weather, the time of the day, and *interactional*, not directly observable - user's current shopping intent, his current mood. Context is particularly important for systems where there are many new or anonymous users (such as online news portal). Since no historical data is available about users' past preferences, it is very important to make full use of interactional context information to achieve the best possible personalization.

Trend detection. Since the popularity of items changes over time, moreover, in news domain items become outdated very fast, sequence-aware recommenders aim to detect and utilize popularity patterns over time to improve recommendations. Trends can be long-term, seasonal, or short-term. What is more, trends can be individual, or representing the different communities. As it is shown in paper "Personalized News Recommendation based on click behavior" by Liu et al. [13], individual user interests change over time and are influenced by global and local trends, in other words, individuals follow the public.

3.5.2 Categorization based on context adaptation type

There are three general approaches used in sequence-aware systems for next-click prediction: session-aware, session-based and last-interaction adaptation [20].

- *Last-N interactions recommendation.* Only the very last user actions are considered when determining the next object to be recommended (e.g., next location recommendation).
- *Session-based recommendation.* The last sequence of user actions is known and this sequence of actions is limited to a session (e.g., news recommendation, e-commerce, video and classified advertisement recommendation).
- *Session-aware recommendation.* Applied when we have knowledge both about user's actions in the last session and about his past behavior (returning customers). Such systems can be based on a combination of long-term and short-term interest models (e.g., e-commerce, app recommendations).

Most of the research works (by the statistics provided in the recent comprehensive survey of sequence-aware recommender systems [20] published on Feb 2018) is done on last-interactions adaptation. A slightly less researched are session-based recommendation techniques, and even less attention is paid to session-aware recommenders, although a noticeable increase in scientific interest is identified - more than half of the papers on session-based and session-aware recommendation were published after 2014.

3.6 News Application Domain

This subsection is dedicated to the news recommendations topic, as this is the domain for the theoretical research and experimental part of this thesis.

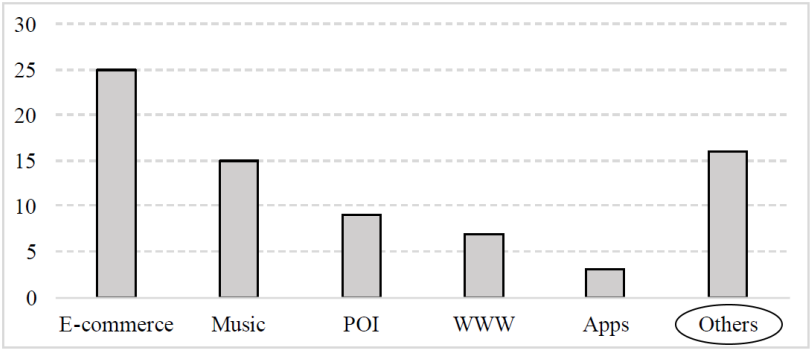


Figure 1: Distribution of research works per application domain adopted from [20]

As shown in Figure 1, presented in the previously mentioned survey on sequence-aware RS [20], the News domain does not appear among the 5 most widely researched next-item recommendation application domains, and falls into the category "Others", together with advertisement, job, query, video in streaming platforms recommendations, and others. This acknowledges the fact that news recommendation domain is not adequately explored yet, and a lot of analytical and experimental work is still to be done.

News RS have specific characteristics and particular challenges. We further discuss them in detail. These challenges apply also for other application domains, but usually in smaller degree.

Data sparsity. One of the distinctive characteristics of news application domain is its especially dynamic environment, where hundreds of new articles are published every day, and get outdated very quicky - usually in 2-7 days. Each user is able to experience just a very small portion of available articles, making it hard to build meaningful models and conclusions about their preferences, while a lot of preferences are left unknown.

Lack of users personal information. For building long-term user profiles, the problem is that readers tend to consume online news without authentication [4], thus each time system treats them as new ("one-time") users, and the long-term profile cannot be updated. "Cookies" are used to reduce the problem, but the cross-device and cross-browser tracking still stays technically challenging issue. In many existing news recommenders, the most popular and most recent news are recommended (unpersonalized recommmendation approach), resulting in good accuracy and outperforming many other state-of-the-art algorithms, that show excellent results on other application domains when relying on availability of rich user profiles. Moreover, the news readers most of the times desire to read breaking news that are independent from their general interests [6], to stay informed. Besides, even if owning enough data about historical user interactions, their interests are changing over time very fast, comparing to other domains: general user interest in movie or music genre is more stable than in news category.

Implicit feedback misinterpretation. When dealing with implicit user feedback like clicks and views, it is difficult to investigate actual user attitude towards the items he clicks on. Often user actions might be *unintended* [14] (a click by accident, a click made by a child when playing, etc.) or a user might realize after spending some time on analyzing the item that it is not what he was looking for. Recently, the later problem is tried to be solved with asking users direct questions like "Was this article useful?", but vast majority of users tends to refuse to give explicit feedback, usually ignoring the question.

The following challenges are more common for all application domains, but are included in this section for simplicity and to ease the reading.

Recommendation bias. User behavior, in modern online production systems, is - at least expectedly - influenced by recommendations themselves. It is reasonable to assume, for instance, that recommended items will be more likely followed than if they were not recommended. Simulating this offline usually requires complicated user behavior modeling which can be expensive and prone to systematic error [26]. Thus, when performing offline model evaluation and trying to reach the best accuracy may lead to a paradox - instead of recommending items that would fit user intentions the most, we recommend items that were actually recommended to the user in the past to result in accurate predictions.

Users' "right to explanations". On 25 May 2018 an European Union General Data Protection Regulation (GDPR) becomes enforceable. Together with other user rights, like "right to access" and "right to erasure", it involves a "right to explanation" of algorithmic decisions. This new regulation has a major effect on recommender systems, as users get a right to ask "Why you recommend me this?". Simply giving access to underlying code is insufficient for most people, moreover, this would most likely be also unacceptable to the owner of the code. This potentially prohibits a wide swath of algorithms currently in use, and requires a complete overhaul of the machine learning community accepted standards, while their used approaches most often lack interpretability [5]. There is a mismatch between the mathematical optimization in high-dimensionality characteristic of machine learning and the demands of human-scale reasoning and styles of interpretation. New techniques must be developed that can not only make more accurate predictions, but offer increased transparency and fairness.

4 Graph-based similarity measures

While most of the traditional recommendation algorithms rely either on a content-based approach or a collaborative filtering, the graph-based recommender systems naturally combine the two approaches, as all the information about both user-item interactions and their descriptive features lays in one graph structure, where participating units (users, items, features) are represented by graph nodes, and their relations - as graph edges.

However, up to now, graph-based algorithm applications for recommender systems (or, more specifically, - for sequence-aware recommender systems) did not attract proper attention, and as shown in taxonomy of algorithms for SARS, presented in an already mentioned recent survey [20], are classified into "Others" with just some research work examples.

Fortunately, a modern tendency promises changes to this discrimination, and in the last *The Web Conference* (formerly WWW conference) held in April 2018, quite a lot of attention was paid to the graph algorithms that can be used in recommender systems, social network analysis, web content analysis and semantics extraction, web mining, user modelling, etc. [1].

In this chapter we present the concept of a heterogeneous information network, which is a more commonly used term in recommender systems research domain, and provide a network schema of an online news portal, which is further used in this work for both toy examples and experiments on real data. Then, we discuss the three most commonly used graph-based similarity search algorithms, in particular, SimRank, Personalized PageRank (also known as Random Walk with Restart), and PathSim. We also propose a new measure PathCount, that is supposed to solve drawbacks of former algorithms and is assumed to be more suitable for news recommendation domain.

4.1 Heterogeneous Information Network

DEFINITION 1. ***Information Network.*** *[27] An information network is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with an object type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{Q}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where each object $v \in \mathcal{V}$ belongs to one particular object type $\phi(v) \in \mathcal{Q}$, and each link $e \in \mathcal{E}$ belong to a particular relation $\psi(e) \in \mathcal{R}$.*

Differently from the traditional network definition, we explicitly distinguish object types and relationship types in the network. When the types of objects $|\mathcal{Q}| > 1$ or types of relations $|\mathcal{R}| > 1$, the network is called **heterogeneous information network**.

EXAMPLE 1. A network for online news portal is a heterogeneous information network, containing objects from five types of entities $\mathcal{Q} = \{U, S, A, C, L\}$: users ($U$), sessions ($S$), articles ($A$), article categories ($C$), and article locations (L). Each user $u : \phi(u) = U$ has one or more links to the sessions $s : \phi(s) = S$, each session s has one unique user u associated with it and one or more articles $a : \phi(u) = A$ read by the user within the session. Finally, each article a can appear in one or more sessions s, belong to one news category $c : \phi(c) = C$ and be assigned to one geographic location $l : \phi(l) = L$.

4.2 Network Schema

DEFINITION 2. ***Network Schema.*** *[27] The network schema is a meta template for a heterogeneous network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the object type mapping $\phi : \mathcal{V} \rightarrow \mathcal{Q}$ and a link type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$, which is a directed graph defined over object types $\mathcal{Q}$, with edges as relations from $\mathcal{R}$, denoted as $T_G = (\mathcal{Q}, \mathcal{R})$.*

Network schema serves as a template for a network, and tells how many types of objects there are in the network and where the possible links exist. Network schema for online news portal is shown in Figure 2. A more detailed representation, defining also node and edge attributes and relationship types, is presented in Figure 3.

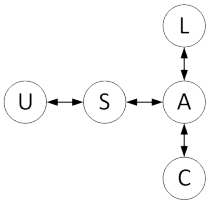


Figure 2: Online News portal Network Schema

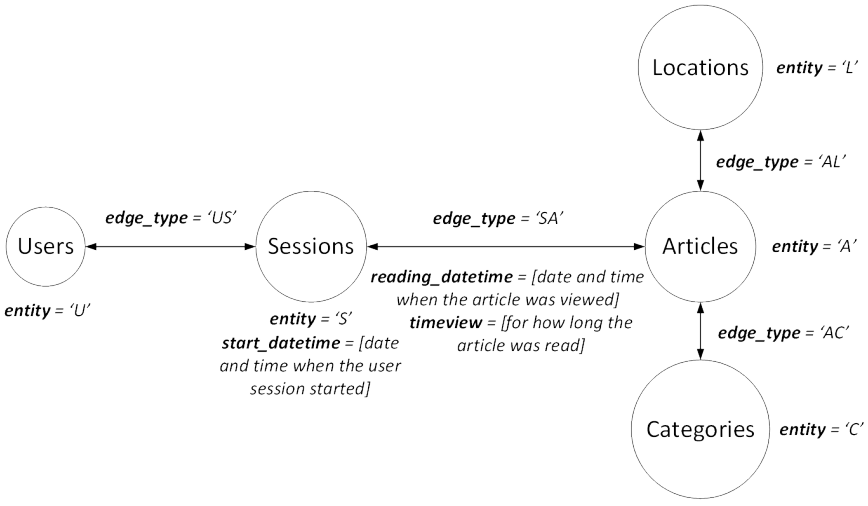


Figure 3: Online News portal Network Schema (detailed)

4.3 SimRank

In 2002 Jeh and Widom [11] first proposed a complementary approach, applicable in any domain with object-to-object relationships, that measures similarity of the structural context in which objects occur, based on their relationships with other objects. The measure can be used to measure similarities of nodes from one enitity (e.g. similarities between users by considering their friends), from two entities (e.g. similarities between users by considering items they interacted with).

For different recommendation approaches (e.g. Content-based, Collaborative filtering, Session-based), different general SimRank statements apply, though all corresponding to the underlying idea, that *two objects are similar if they are related to similar objects*. In Table 2 the SimRank statements for three discussed approaches are presented.

Case	SimRank statement
Base case	Two objects are similar if they are related to similar objects.
CF case	Two items are similar, if they are read by similar users, Two users are similar if they read similar items.
CB case	Two items are similar if they belong to similar categories, Two categories are similar if they unite similar items
SB case	Two items are similar if they appear in similar sessions, Two sessions are similar if they hold similar items.

Table 2: SimRank similarity statements for different recommendation cases

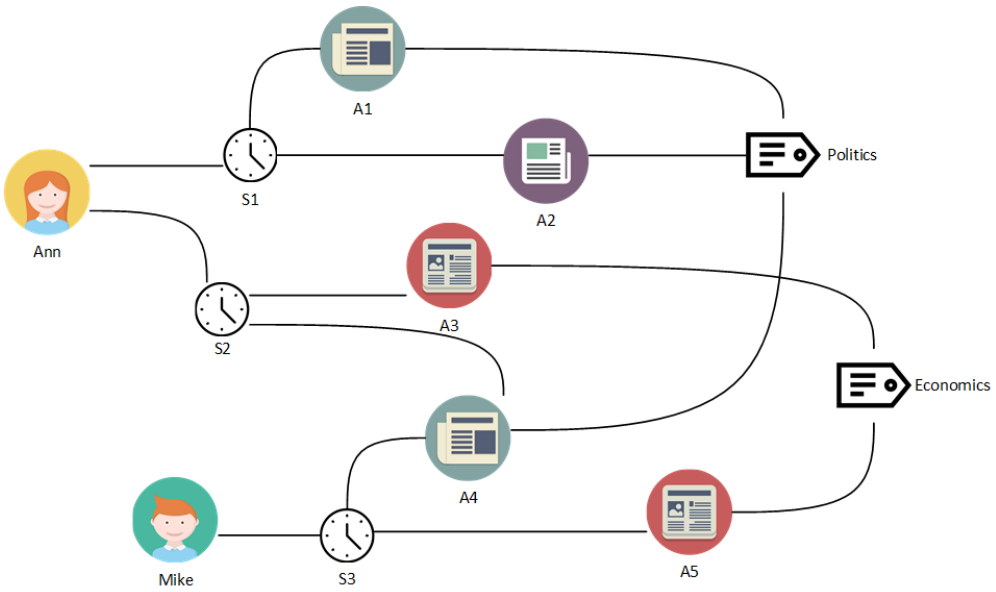


Figure 4: Toy example of a heterogeneous graph network USAC

DEFINITION 3. ***SimRank.*** *A SimRank similarity score between two graph vertices a and b is defined as*

$$s(a,b) = \begin{cases} 1, & \text{if } a = b; \\ \frac{C}{|N(a)||N(b)|} \sum_{i=1}^{|N(a)|} \sum_{j=1}^{|N(b)|} s(N_i(a), N_j(b)), & \text{if } a \neq b, N(a) \neq \emptyset, N(b) \neq \emptyset; \\ 0, & \text{if } N(a) = \emptyset \text{ or } N(b) = \emptyset, \end{cases} \quad (1)$$

where $C \in [0,1]$ - the damping factor, $N(a)$ - neighbors of item a, $N(b)$ - neighbors of item b.

EXAMPLE 2. Let's consider a simple toy example of a heterogeneous graph with a structure USAC (user-session-article-category), shown in Figure 4. From the following graph structure we can extract different sub-graphs, corresponding to different recommendation approaches (see Table 3). We then iteratively apply SimRank function (Equation 1) on different sub-graphs, calculating article-article similarities, using damping factor $C = 0.8$ and with the convergence criteria $\varepsilon \leqslant 10^{-4}$. Table 4 shows the resulting similarity scores for A1 with four other articles.

Sub-graph	Approach
AC	Content-based (CB)
UA	Collaborative filtering (CF)
SA	Session-based (SB)
USA	CF+SB
SAC	SB+CB
USAC	CF+SB+CB

Table 3: Recommendation cases corresponding sub-graphs of the USAC network

	CB	CF	SB	CF+SB	SB+CB	CF+SB+CB
A1-A2	0.8	0.8	0.8	0.8	0.608	0.579
A1-A3	0	0.8	0	0.281	0.171	0.202
A1-A4	0.8	0.573	0	0.198	0.343	0.342
A1-A5	0	0.347	0	0.115	0.171	0.167

Table 4: SimRank similarities of article A1 with other articles on different sub-graphs

CB. Sub-graph: AC. Only articles A2 and A4 belong to the same category, as article A1, thus they get similarity score of 0.8 (our damping factor C). Other articles are treated as absolutely dissimilar.

CF. Sub-graph: UA, derived from the original graph by joining users with articles directly, skipping the sessions grouping (traditional CF case). An observation here is that even though articles A1 and A5 are not read by the same user, they get a similarity score of 0.347, because they are read by *similar* users, which in their turn become similar because they have read the same article A4.

SB. Sub-graph: SA. Only article A2 appears in the same session with A1, thus all other articles are treated as dissimilar. *Note:* Even though articles A3 and A5 do not belong to the same session, they would get a similarity bigger than 0 (not shown in the Table 4), because sessions they belong to are similar - both hold an article A4.

CF+SB. Sub-graph: USA. The pair A1-A2 has the same similarity as for single CB or CF approach, as they both are read by one user (Ann) within the same session. A1-A3 get a lower similarity, as they are read by the same user but within different sessions (note, that the time interval between those sessions could be 1 month or 1 year, and so those articles would very unlikely be similar). Another important observation is, that even though A4 is read together with A3 and seems to have to receive the same similarity with A1, their similarity is decreased, because A4 is also read by another user (Mike), therefore becomes less relevant to Ann.

SB+CB. Sub-graph: SAC. A1-A2 again get the biggest similarity, because of belonging to the same session and category. A4 gets lower similarity, as only belongs to the same category, but different session. Articles A3 and A5 get similarity to A1 bigger than 0, even while belonging to different categories and different sessions. The reason is, that they both co-appear in a session together with A4, which is similar to A1, which makes sessions S1, S2 and S3 also similar, and finally, their articles A3 and A5 also similar to A1.

CF+SB+CB. Sub-graph: USAC. The approach, that captures all available information and is the best representative of the real-world. The applied logic is very similar to the previous SB+CB case, with the only difference, that the pair A1-A3 wins against A1-A5, because of being read by the same user Ann (sessions S1-S2 become more similar than S1-S3).

4.4 Personalized PageRank/Random Walk with Restart

The idea of Personalized PageRank comes from the famous PageRank algorithm [18], proposed back in 1998, and used for measuring the importance of Web pages on the network by exploring links between them. The algorithm relies on the idea of abstract "random surfers" (a.k.a. "random walkers") that start at an arbitrary node of the network and "walk" through the links, sometimes arbitrarily jumping to another random node of the network. As a result, the more often the "surfer" passes through a node, the more important that node is in the network.

Similarly, for social graphs with different types of nodes, we can imagine a "random walker", that from any node N moves to any neighbour of N with equal probability, at each step having a constant probability to return back to the initial node. This way we want to identify where the random walker winds up on short walks starting from a target node, i.e. which network objects are the most strongly related (the most similar) to the target one. The similarity measure, calculated this way, is referred to as **Personalized PageRank** (P-PageRank) [18], Topic-Sensitive PageRank [7], or a **Random Walk with Restart** (RWR) [12], by the name of the algorithm used for inferring similarities. There is no clear consensus reached about the name to be used. In this report we most often stick to the RWR shortening, as it seems to the author the most convenient and representative name, standing for both the algorithm and the corresponding similarity measure.

As a result, after performing RWR on a graph with n nodes we get an $n \times n$ matrix V, where each element V_{ij} corresponds to a relative importance of a node i to node j (differently from SimRank, where similarities are only inferred between nodes of the same type). Moreover, in contrast to SimRank similarity matrix, the P-PageRank/RWR matrix is not symmetric - a relative importance of a node i towards node j is not equal to the importance of a node j towards node i (it is more likely to end up in a popular object when starting from a non-popular, rather than other way around).

Let's define $\mathbf{M}$ as the initial transition matrix of the graph G. That is, the entry in row i and column j of $\mathbf{M}$ is $1/k$, where k is the node degree of j from G, and i is one of it's adjacent nodes. If $k = 0$, this entry is 0. Let's also define the probability β for a "walker" to continue at random, and, consequently, $1 - \beta$ - probability to jump to the initial node N. Let $\mathbf{e_N}$ be the column vector with 1 in the row for node N and 0's elsewhere. Then, if $\mathbf{v}$ is the column vector that reflects the probability the "walker" is at each of the nodes at a particular round, and $\mathbf{v}'$ is the probability the walker is at each of the nodes at the next round, then $\mathbf{v}'$ is related to $\mathbf{v}$ by:

$$\mathbf{v}' = \beta \mathbf{M} \mathbf{v} + (1 - \beta) \mathbf{e_N} \tag{2}$$

The Equation 3 transforms the Equation 2 to be applied on the whole network, computing an $n \times n$ transition probability matrix for any pair of nodes by a power iteration method.

DEFINITION 4. ***Random Walk with Restart.*** *A non-symmetric, row-normalized transition matrix* $\mathbf{V}'$*, storing transition probabilities between any two nodes of the information network, can be calculated iteratively by using the following formula:*

$$\mathbf{V}' = \beta \mathbf{M} \mathbf{V} + (1 - \beta)\mathbf{I_n}, \tag{3}$$

where, $\mathbf{I_n}$ *is an* $n \times n$ *identity matrix, and* $\mathbf{V}$ *and* $\mathbf{V}'$ *are column-normalized transition probability matrices in two consecutive rounds.*

Let's return to our toy example (Figure 4) and run on it the RWR algorithm. The main advantage of the result matrix is that it consists of meaningful sub-matrices (see Table 5), useful for different purposes. For example, the Article-Article sub-matrix (the third from the top and from the left) shows article-article transition probability matrix[1], and can be directly applied for next-click prediction given the currently viewed article (classical Markov process). The User-Category sub-matrix (right upper corner), on its turn, reveals general tastes of users by defining how probable it is for the user to read an article from one or another category. Similarly, Session-Article sub-matrix (the second from the top and the third from the left) can help with direct next article recommendation, by proposing the article with the highest probability score at the session row, from those that are not yet viewed within that session.

	Ann	Mike	S1	S2	S3	A1	A2	A3	A4	A5	Politics	Economics
Ann	0.284	0.009	0.163	0.153	0.033	0.061	0.061	0.052	0.067	0.021	0.067	0.029
Mike	0.018	0.276	0.019	0.047	0.287	0.016	0.016	0.034	0.099	0.097	0.039	0.052
S1	0.108	0.007	0.339	0.067	0.025	0.119	0.119	0.024	0.054	0.012	0.110	0.014
S2	0.102	0.016	0.067	0.314	0.059	0.033	0.033	0.107	0.115	0.039	0.057	0.059
S3	0.022	0.096	0.025	0.059	0.358	0.019	0.019	0.042	0.124	0.122	0.049	0.065
A1	0.061	0.008	0.179	0.049	0.029	0.294	0.094	0.018	0.067	0.013	0.173	0.012
A2	0.061	0.008	0.179	0.049	0.029	0.094	0.294	0.018	0.067	0.013	0.173	0.012
A3	0.052	0.017	0.036	0.161	0.063	0.018	0.018	0.304	0.068	0.078	0.033	0.153
A4	0.045	0.033	0.054	0.115	0.124	0.045	0.045	0.046	0.294	0.048	0.114	0.037
A5	0.021	0.049	0.019	0.059	0.183	0.013	0.013	0.078	0.072	0.311	0.029	0.155
Politics	0.045	0.013	0.110	0.057	0.049	0.116	0.116	0.022	0.114	0.019	0.033	0.017
Economics	0.029	0.026	0.022	0.088	0.098	0.012	0.012	0.153	0.056	0.155	0.025	0.323

Table 5: Random Walk with Restart result transition matrix

[1]Note, that in the Table 5 we show a transposed result matrix, which is row-normalized, and the element (i,j) defines the transition probability from node i to node j.

An interesting and not so intuitive observation can be noticed in the User-Category sub-matrix, where we can see that even though Mike has read one article from each category (politics and economics), his interest is biased towards economics. The reason is, that article A4, read by Mike in session S3, appears in the Ann's session S2 together with A3, which is a economics article, thus bringing A4 closer to economics category, and consequently, increasing Mike's assumed interest towards economics.

What is more, for the news application domain, the short-term user preferences as well as short-term article-article and category-category relations should be considered more than long-term general object relationships. This can be done by removing sessions older than a specified time stamp from the graph, together with articles that don't appear in any latest sessions. We discuss this property in more detail in Chapter X, when describing the proposed model and prequential evaluation protocol.

In the experimental part we compare the results of running the algorithm on different sub-graphs and on differently defined model learning terms.

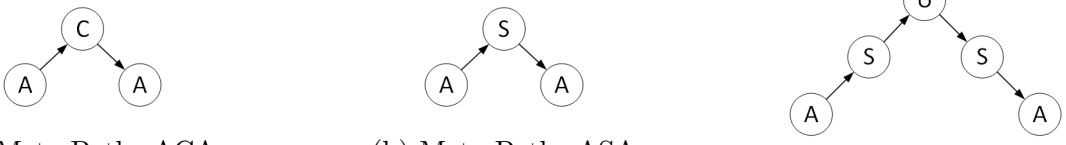
4.5 PathSim

In 2011 Sun et al. [27] proposed a novel idea of measuring similarities between network objects by analyzing meta-paths through which objects are connected. In a heterogeneous network, two objects can be connected through different paths. For example, to articles can be connected through the path "article-category-article" (content based similarity), "article-session-article" (session based similarity), "article-session-user-session-article" (collaborative filtering based similarity). Using different paths, different similarities are observed. These paths are called *meta paths* and are formally defined as follows:

DEFINITION 5. ***Meta Path.*** *[27] A meta path $\mathcal{P}$ is a path defined on the graph of network schema $T_G = (\mathcal{Q}, \mathcal{R})$, and is denoted in the form of $Q_1 \xrightarrow{R_1} Q_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} Q_{l+1}$, which defines a composite relation $R = R_1 \circ R_2 \circ \dots \circ R_l$ between type Q_1 and Q_{l+1}, where $\circ$ denotes the composition operator on relations.*

EXAMPLE 3. There are many various meta paths that can be built on the USACL network. If we start from the article node, we can build the following meta paths: ACA, ASA, ASUSA. (see Figure 5).

Main advantage of the meta path framework is its explainability and controllability, while it provides a powerful mechanism for a user to select an appropriate similarity semantics, by choosing a proper meta path. A similarity measure that is able to capture the semantics of similarity among network objects is called **PathSim**.



(a) Meta Path: ACA (b) Meta Path: ASA (c) Meta Path: ASUSA

Figure 5: Possible Meta Paths of the USAC Network starting from A

DEFINITION 6. ***PathSim: A Meta path-based similarity measure.*** *[27] Given a symmetric meta path P, PathSim between two objects of the same type x and y is*

$$s(x,y) = \frac{2 * |p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in P|}{|p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in P| + |p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in P|}, \tag{4}$$

where $p_{x \rightsquigarrow y}$ is a path instance between x and y, $p_{x \rightsquigarrow x}$ is that between x and x, and $p_{y \rightsquigarrow y}$ is that between y and y.

EXAMPLE 4. Let's explore article similarities using simple meta paths of our toy example from Figure 4: ACA (content-based filtering), ASA (session-based filtering), and ASUSA (collaborative filtering). Table 6 shows similarities of article A3 with other articles. A3 only shares the category with A5, thus $\text{PathSim}_{\text{ACA}}(\text{A3,A5}) = \frac{2*1}{1+1} = 1$. A4 shares sessions with A3 and A5, thus $\text{PathSim}_{\text{ASA}}(\text{A3,A4}) = \frac{2*1}{1+2} \approx 0.666$. Similarly, as A3 and A1 share the same user, $\text{PathSim}_{\text{ASUSA}}(\text{A3,A1}) = \frac{2*1}{1+1} = 1$, and as A3 and A5 do not share a user, $\text{PathSim}_{\text{ASUSA}}(\text{A3,A5}) = \frac{2*0}{1+1} = 0$.

	ACA	ASA	ASUSA
A3-A1	0	0	1
A3-A2	0	0	1
A3-A4	0	0.666	0.666
A3-A5	1	0	0

Table 6: PathSim similarities of article A3 and other articles using different meta-paths

Differently than SimRank and RWR, that are biased towards more popular items in the network, PathSim captures also the *visibility* of objects, bringing closer the objects that share similar visibility. For example, if two researchers have published 10 papers each and the third researcher has published 100 papers, SimRank and RWR would make two "less active" researchers very similar to the "active" one, and less similar to each other, while the PathSim would reduce the similarity with the "active" one and bring two "not active" researchers close to each other, conceptually considering their "activeness" as yet another similarity factor (new unexperienced researchers are more similar to each other than to the acknowledged experts of the field).

4.6 PathCount

Even though PathSim similarity measure is widely used and recognized, to our belief it is not the best option for the news application domain. Considering visibility of objects is useful in many domains, but in news domain, where actuality and popularity of articles plays a meaningful role, punishing articles for their popularity, in our opinion, is not fair. Recommending only the news of the same visibility may lead to improper reader information about the actual situation and state of affairs. Therefore, we propose a simplification of the PathSim method, where we measure only the number of paths through which two nodes are connected (PathSim's numerator), and normalize the measures to result in transition probabilities for each node.

DEFINITION 7. ***PathCount: A Meta path-based transition probability measure.*** *Given a symmetric meta path P, PathCount between two objects of the same type x and y is*

$$s(x,y) = \frac{|p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in P, x \neq y|}{\sum_{z \in \mathcal{V}: \phi(z) = \phi(x)} |p_{x \rightsquigarrow z} : p_{x \rightsquigarrow z} \in P|}, \tag{5}$$

where $p_{x \rightsquigarrow y}$ is a path instance between x and y, and $p_{x \rightsquigarrow z}$ is that between x and z, where z is any object of the same type as x.

EXAMPLE 5. Let's consider a simple bipartite session-article graph with 3 articles and 5 sessions (see Figure 6). Each pair of articles is connected through one common session S3. Additionally, article A2 appears in 4 more sessions - is very popular in the network. As we can see from the Table 7, the PathSim measure punishes article A2 for its popularity and significantly reduces their similarity with A1. Random Walk with Restart, being strongly biased towards popular items, on the contrary, makes pair A1-A2 closer to each other, because a random walker on his way much more often passes though A2 than A3, even though his walk is not related to the source article A1 itself. Our proposed PathCount measure treats two articles equally relevant for A1, because they all appear in the only one common session and have equal rights to be read next.

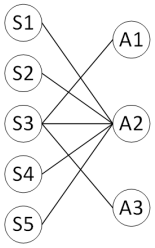


Figure 6: Bipartite session-user graph toy example

	PathSim	PathCount	RWR
A1-A2	0.33	0.5	0.18
A1-A3	1.0	0.5	0.08

Table 7: RWR, PathSim, PathCount measures for article A1 with A2 and A3

5 Building recommendations

In the previous chapter we discussed the four graph-based similarity measures. In this chapter we describe how we build recommendations using on one or another similarity measure. We firstly propose the four approaches of capturing user's short-term intention to find the articles that are fitting best to be recommended next. For this purpose we use only item-item relevance matrix. Next we go beyond short-term intentions and propose a way to capture also medium-term and long-term object relationships, such as category-category transitions and user-category preferences. Finally, we describe a Session-kNN method, that makes use of session-session similarities.

As long as similarity search algorithms discussed in the previous chapter provide different types of outcomes (similarity matrix, transition probability matrix), further we refer to terms "relevance matrix" and "relevance vector" assuming any kind of relationship between a pair of nodes.

5.1 Weighting session articles

In this section, we compare four different ways of capturing the importance of each article in a user session to infer his short-term reading goals. As discussed in [15], user preferences might change within even a short interaction with a system depending on the already seen items. Thus, we want to identify how we should weight the articles of the session to best reflect user current preference. Here are the four approaches to be compared:

1. $\mathbf{W_{LA}}$ - *only the last viewed article matters.* The recommendation is made by finding the most relevant articles to the *currently viewed one.* This approach provides non-personalized recommendations, as only relies on one item. This approach is most commonly used in modern e-commerce systems, such as *Amazon*, *eBay*, etc., where the recommended items are divided into groups like: "This item was frequently viewed together with ...", "People who viewed this item also viewed ...", etc.;

2. $\mathbf{W_M}$ - *all articles of the session are equally important.* The simple *mean* of relevance vectors of all articles revealed from the session is calculated, and those articles are recommended, that have the highest average similarity score - are the most relevant to all user session;

3. $\mathbf{W_S}$ - *the closer the article to the end of the session, the more important it is.* We suppose, that recommended articles should be the most relevant to the most recent

session articles, and the least relevant to the oldest ones. We propose to use a Sigmoid-based weight assignment function.

The general Sigmoid function is stated as:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

As long as Sigmoid lays in the interval $\approx [-5; 5]$, where $S(-5) \approx 0$, $S(5) \approx 1$, we propose to assign weights of the articles in a session using the following equation:

$$w(a) = S(-5 + \frac{10 * n(a)}{N}), \tag{7}$$

where $n(a)$ - position of the article in a session, N - total number of session articles.

The weights are then normalized by dividing each by the sum of all, so that all weights sum up to one.

4. $\mathbf{W_T}$ - *the longer is the reading duration of an article, the more important it is.* A large amount of articles within user sessions are read for less than 20 seconds, usually denoting either unintended user actions or a lack of user interest in the article. Therefore, we propose to make the weight of an article depend on its timeview. We use the following piecewise logarithmic function to bring closer articles that were read for long time enough and give very small weights to not interesting articles:

$$w(a) = \begin{cases} log_{10}(t(a) + 1), & \text{if } t(a) < t_{breakpoint}; \\ ln(t(a) + 1), & \text{if } t(a) \geqslant t_{breakpoint}, \end{cases} \tag{8}$$

where $t(a)$ is the time duration of the article a within the session in seconds, and $t_{breakpoint}$ is a time duration under which we treat the article views unimportant. In our experiments we use $t_{breakpoint} = 20$. Normalization of weights is performed afterwards.

EXAMPLE 6. Assume, that a user have just read two articles $[a_1, a_2]$ one after another. They were read for 120 and 10 sec correspondingly. From the learned recommendation model, based on any of the known similarity measures, we know the relevance vectors for both articles:

Let's analyze the results obtained by using four proposed strategies with recommending 2 most relevant articles, i.e. $_{TOP}N = 2$, presented in Table 9.

All four methods produced different results. TOP 1 recommendation for W_{LA} and W_S is the same, because both methods are biased towards article a_2 as the last read

	a_3	a_4	a_5
$rel(a_1)$	0.1	0.7	0.2
$rel(a_2)$	0.6	0.1	0.3

Table 8: Item relevance vectors for items $a1$ and $a2$

Strategy	Weights $[a_1, a_2]$	a_3	a_4	a_5	Top-2 Rec
W_{LA}	[0, 1]	0.6	0.1	0.3	a_3, a_5
W_M	[0.5, 0.5]	0.35	0.4	0.25	a_4, a_3
W_S	[0.33, 0.67]	0.43	0.3	0.27	a_3, a_4
W_T	[0.82, 0.18]	0.19	0.59	0.22	a_4, a_5

Table 9: Comparison of weight assignment strategies

article. However, the second recommended article is different: W_{LA} recommends simply the most relevant to a_2, while W_S considers a_4 to be more relevant, because it has a very high relevance to the previously read article. W_T, on its turn, is biased towards firstly read article a_1, because it was being read for two minutes, while the a_2 was only read by 10 sec, i.e. probably was not interesting to the user. W_M recommends just articles that have the biggest average relevance score.

5.2 Combining short-, medium- and long-term interests

In the previous section the personalization of recommendations was limited by user's current session, i.e. his short-term intentions. The first to propose considering medium- and long-term crowd reading interests together with short-term interests was Epure et al. in 2017 [4], where she combined the popularity baseline of short-term crowd behavior with either medium-term or long-term category-category transition probability matrix, and found out that combination of short- and long-term interests result in increased accuracy, while combination of short- and medium-term interests yield higher news variety.

In this section we propose a new idea to combine together short-, medium- and long-term object relationships. We consider item-item relevance matrix on the short-term level (e.g. "Which articles were recently read together with the ones from current user session?"), category-category transitions on medium-term level (e.g. "According to the last week information, what is the probability to read politics article after economics one?"), and user-category preferences on the long-term level (e.g. "What is the favorite news category of the given user?").

In the Table 10 we refer to S as short-term, M - as medium-term, and L - as long-term.

Approach	Sub-graphs	Comments
S	AA	Only article-article relationships matter at short-term period
SM	AA+CC	We consider recent relationships between articles as well as the transition probabilities between their categories learned on medium-term period
SL	AA+UC	We consider recent relationships between articles as well as users preferences towards potentially recommended article categories learned on long-term period
SML	AA+CC+UC	All relationships are considered together

Table 10: Short/Medium/Long-term based model settings

S. Short-term aware recommendations. This approach covers the strategy discussed in the previous section. The article-article matrices are learned and the recommendations are made based on them.

SM. Short- and medium-term aware recommendations. Within this approach we not only evaluate article-article similarities, but also take into account categories that the articles belong to. Formally, the new relevance $rel_{SM}(u,i,j)$ of item j towards item i for the user u, calculated using SM approach can be found using the following formula:

$$rel_{SM}(u,i,j) = w_S \cdot AA(i,j) + w_M \cdot \mathrm{CC}(cat(i),cat(j)) \tag{9}$$

SL. Short- and long-term aware recommendations. The approach follows similar logic to that from SM, with the only difference, that instead of medium-term category-category matrix the long-term user-category matrix (UC) is learned. The item relevance vector is then multiplied by the scores of current user interest in the given news category.

The formula for calculating $rel_{SL}(u,i,j)$ is very similar to the one before:

$$rel_{SL}(u,i,j) = w_S \cdot AA(i,j) + w_L \cdot \mathrm{UC}(u,cat(j)) \tag{10}$$

SML. This is a combined method of SM and SL, where all three matrices are considered. The formula for calculation:

$$rel_{SL}(u,i,j) = w_S \cdot AA(i,j) + w_M \cdot \mathrm{CC}(cat(i),cat(j)) + w_L \cdot \mathrm{UC}(u,cat(j)) \tag{11}$$

This is a general formula, and can be transformed to S, SM or SL approaches by setting w_M, w_L or both to 0.

EXAMPLE 7. SM Approach. Assume, a user U1 has just read article A5. You have learned the item relevance vector for it ($AA(A5, j)$ in Table 7a). The most relevant article to A5 is A1, and would be recommended in case of the S approach. You have also learned the category-category relevance matrix (Table 7b). The final relevance vector is calculated as a weighted mean of AA and corresponding CC scores with $w_S = w_M = 0.5$. The most relevant article appears to be A2 and is recommended within SM approach.

$AA(A5, j)$		**A1**	A2	A3	A4
	A5	**0.4**	0.2	0.1	0.3
$CC(cat(A5), cat(j))$		C2	C1	C1	C2
	C1	0.2	0.5	0.5	0.2
$rel_{SM}(U1, A5, j)$		A1	**A2**	A3	A4
	A5	0.3	**0.35**	0.3	0.25

(a) Item relevance vector calculation procedure for SM-based recommendations with $w_S = w_M = 0.5$

	C1	C2	C3
C1	**0.5**	**0.2**	**0.3**
C2	0.1	0.6	0.3
C3	0.2	0.3	0.5

(b) Category-category transition matrix learned on medium-term training set

Figure 7: Short- and medium- term aware recommendation schema

5.3 Session-kNN

The Session-kNN method for next item recommendation was first proposed by D. Jannah in 2017 [9], who have shown that a simple k-nearest neighbor scheme adapted for session-based recommendations outperforms other algorithms. Formally, the proposed method is defined as following:

DEFINITION 8. ***Session-based kNN.*** *[9] The kNN method takes the set of user actions in the current session, e.g. two view events for certain items, and then in a first step determines the k most similar past sessions in the training data. Then, given the current session s, the set of k nearest neighbors N_s, and a function $sim(s1, s2)$ that returns a similarity score for two sessions $s1$ and $s2$, the score of a recommendable item i is*

$$score_{KNN}(i,s) = \sum_{n \in N_s} sim(s,n) \times 1_n(i), \tag{12}$$

where $1_n(i) = 1$ if n contains i and 0 otherwise.

The similarity measure used by Jannah in experiments is cosine similarity, as it was found out that the best results are achieved when encoding sessions as binary vectors of the item space. In the experimental part of this work we are using graph-based similarity measures for Session-kNN on different sub-graphs and meta paths, and compare their performances at different model parameters, like size of training data, minimum number of revealed items from the session, and others.

6 Explainable recommendations

In this chapter we discuss general explanation styles, propose the framework for explaining news recommendations using meta path based explanations, and the ways of combining several meta paths to achieve better recommendation accuracy and enhance explainability.

6.1 Explanation styles

Receiving an explanation makes a user more easily accept the recommendation, because the system provides transparency and thus gains user trust. The combination of a recommendation along with the reasoning behind it is known as *explained recommendation*.

There are three generally known fundamental resources used for explaining recommendations: *users*, *items* and *features*. In this work we add also the fourth resource - a *session*, which is yet barely used in working systems.

As defined in [19], explanation styles can be classified into: (i) *Human* Style, which provides explanations based on similar users, (ii) *Item* Style, which is based on choices made by users on similar items, and (iii) *Feature* Style, which explains the recommendation based on item features (content). We define the fourth type: (iiii) *Session* Style, which is based on the concept of co-occurence of items in one user session.

In this work we propose three recommendation strategies: **item-based** (IB), **session-based** (SB) and **user-based** (UB), and for each of them define explanation styles that can be used (see Figure 8). Explanation styles can be used both individually for each recommendation strategy or in combinations with each other.

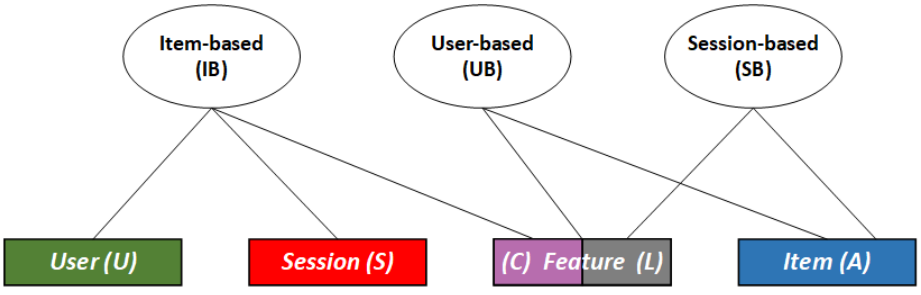


Figure 8: Explanation styles for three recommendation strategies

Referring to the news portal network structure, presented in Figure 2 of Section 4, we specify network meta-paths that are used in each explanation style for each recommendation strategy. As you can see in Table 11, for different recommendation strategies the same explanation styles use different meta-paths.

	IB	SB	UB
Human	A**U**A	-	-
Session	A**S**A	-	-
Item	-	S**A**SA	U**A**UA
Feature (C)	A**C**A	S**C**SA	U**C**UA
Feature (L)	A**L**A	S**L**SA	U**L**UA

Table 11: Meta-paths used in three recommendation strategies

The main ideas of each recommendation strategy are summarized as follows:

- **Article-based.** We start with an article of the current user session and find similar (relevant) articles according to a specified meta-path along a short time period. We recommend articles that are the most relevant. Example of ASA-based explanation: *We recommend you A207, because it co-occurs with your currently viewed article in 10 other user sessions*;
- **Session-based.** We start with the current user session and search for similar sessions according to a specified meta-path along a short time period. We then recommend articles that appear in similar sessions, ranking them by the number of their occurrences (we take a *majority vote*). Example of SCSA-based explanation: *We recommend you A407, because it appears in 12 other user sessions that hold articles from the same category as the articles of your current session*;
- **User-based.** We start with the currently active user and search for similar users based on a specified meta-path along the whole history. We recommend articles, that were recently (within the short time period) read by similar users, ranking them by the number of occurrences. Example of ULUA-based explanation: *We recommend you A311, because it was recently read by 4 other users, who normally read articles from the same location as you.*

6.2 Hybrid explanations

In this section we bring meta-paths together to form combined explained recommendations. Consider the following examples.

EXAMPLE 8. Article-based recommendation using hybrid *Human/Feature (Location)* explanation style. Let's say, a user starts his reading session and clicks on the article A107. A possible explained recommendation for him could sound like: *We recommend you an article A250, because: (i) it was viewed by 5 users that also viewed A107 (AUA), and (ii) it belongs to the same location with it (ALA).*

EXAMPLE 9. Session-based recommendation using hybrid *Session/Feature (Category, Location)* explanation style. Let's say, a user has already viewed three articles during his current session. A possible explained recommendation for him is: *We recommend you article A271, because it was viewed in: (i) 5 sessions together with some of the articles of your current session (SASA), (ii) 7 sessions together with articles from the same category as those from your current session (SCSA), and (iii) 8 sessions together with the articles from the same location as those from your current session (SLSA).*

When using combined recommendations, there is no solely correct answer for the question "Which combinations are stronger than the others?". For example, which article to show first: the one, that belongs to the same category, same location, and co-appears in one session; or the one, that co-appears in 8 sessions, but belongs to different category and location? One can think of many different ways of ranking combined recommendations. In next section we propose a one possible hybrid recommendations ranking procedure.

6.2.1 Hybrid recommendations ranking

Consider a SAC (session, article, category) heterogeneous graph in Figure 9, assuming that session S5 is the current session of an anonymous user, for whom we have to provide a ranked list of recommended articles. As shown, a user has already viewed 2 articles (A4 and A5). The question is, which articles of the potential candidates A1, A2 and A3 to recommend, and in which order?

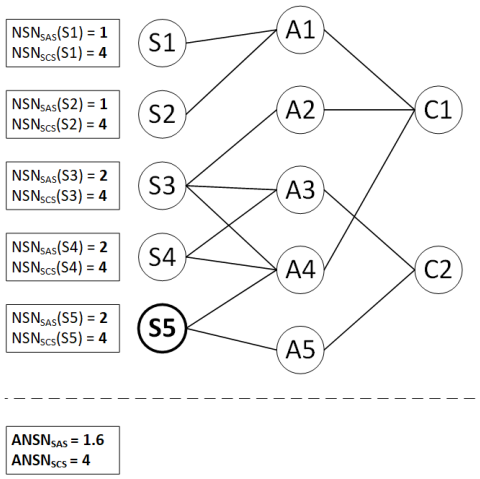


Figure 9: SAC graph toy example for calculating average number of similar nodes in the graph

First, for each potential article we calculate a number of votes ($NV_P(a)$), that denotes in how many similar objects (in our case - sessions) according to a given meta path the candidate item appears. Here we must specify a threshold T for finding similar sessions.

We say Sx is similar to Sy according to meta path P, if $PathCount_P(Sx, Sy) > T$. Table 12 shows the numbers of votes for each candidate item for two meta paths SASA and SCSA, when the threshold $T = 0$.

Article	NV_{SASA}	NV_{SCSA}
A1	0	2
A2	1	1
A3	2	2

Table 12: Number of votes for each candidate article to be recommended based on the number of times that it appears in the meta path connections.

We see, that A3 has $NV_{SASA} = 2$ and $NV_{SCSA} = 2$, which means directly that it is the most strongly connected article to the session S5. However, what is the second relevant article - A1 or A2? What is stronger - 1 SASA and 1 SCSA connecting paths, or 2 SCSA connecting paths? To evaluate this, we measure with how many sessions on average a single session is similar (or is connected, if $T = 0$) given each meta path. We first calculate a number of similar nodes (NSN) for each session, - with how many other sessions each session is similar (using the same threshold T), - and then take the average of them. See again Figure 9 to analyze the values for NSN and ANSN of the graph. More generally, the $ANSN_P$ measure is defined as follows:

DEFINITION 9. ***Average Number of Similar Nodes in a Graph.*** *Given a target entity $E \in \mathcal{Q}$ of graph nodes $\mathcal{V}$, a symmetric meta-path P, and a threshold T, the average number of similar nodes (ANSN) for this meta path is:*

$$ANSN_P = \frac{1}{|x : x \in V, \phi(x) = E|} \cdot \sum_{x \in V, \phi(x) = E} NSN_P(x), \tag{13}$$

where $NSN_P(x)$ stands for the number of similar nodes of a graph node $x \in \mathcal{V}$, $\phi(x) = E$, satisfying the threshold T constraint.

In our running example, the average numbers of similar nodes for two meta paths are: $ANSN_{SAS} = \frac{1+1+2+2+2}{5} = 1.6$, $ANSN_{SCS} = \frac{4+4+4+4+4}{5} = 4$, i.e. each session is *on average* connected with 1.6 other sessions through articles, and with 4 sessions through article categories.

Finally, to rank candidate items for the recommendation list by taking under consideration several meta paths, we calculate a *Candidate Item Relevance*, which is formally defined as follows:

DEFINITION 10. ***Candidate Item Relevance.*** *Given a node $x_{current}$, a list of meta-paths $\mathcal{P} = [P_1, P_2, ..., P_n]$ that form a combined meta paths-based recommendation, and the numbers of votes for article a per meta path P_i ($NV_{P_i}(a)$), the candidate item relevance (CIR) to $x_{current}$ is:*

$$CIR(a) = \sum_{i=1}^{|\mathcal{P}|} \frac{1}{ANSN_{P_i}} \cdot NV_{P_i}(a) \tag{14}$$

In our running example, as shown in Table 13, article A2 has bigger relevance score than A1, because it is has one strong graph connection SASA and one less strong - SCSA, while A1 has two less strong connections. Thus, the final ranked recommendation list is [A3, A2, A1].

Article	CIR
A1	$\frac{0}{1.6} + \frac{2}{4} = 0.5$
A2	$\frac{1}{1.6} + \frac{1}{4} = 0.875$
A3	$\frac{2}{1.6} + \frac{2}{4} = 1.75$

Table 13: Candidate Item Relevance

6.3 Item-based Explained Recommendations

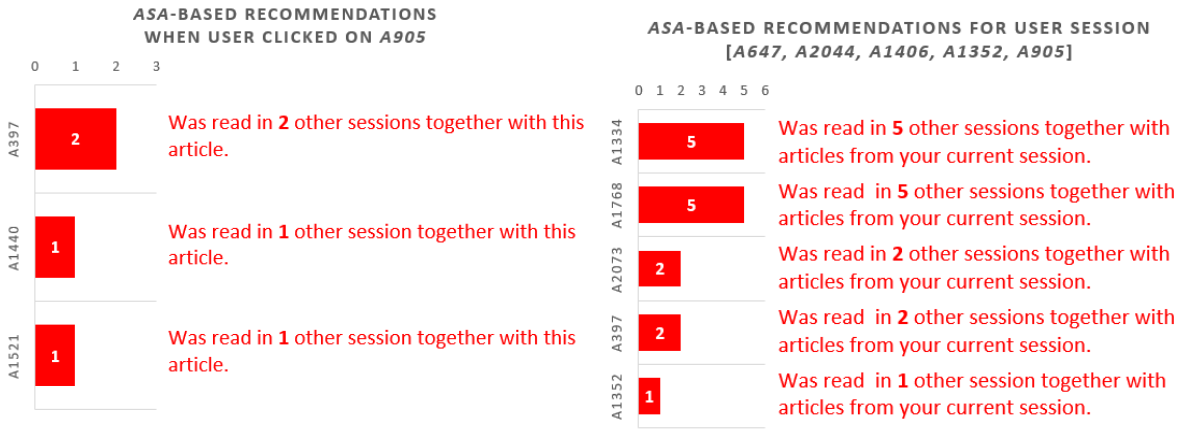
Item-based (IB) explained recommendations use a simple strategy of finding connected (related) articles, and then recommending the most related ones. We distinguish two IB approaches - *Current Item* and *All Session Items*:

1. **Current Item ($IB_{Current}$) Approach.** It is unpersonalized way of recommending next article, where only the currently read article is considered, and the most similar articles to it are recommended;

2. **All Session Items (IB_{All}) Approach.** It is a personalized way of recommending next article, where the whole user session is considered to identify his short-term intention and to recommend articles that best relate to the whole session.

We can use four meta paths: ASA, AUA, ACA and ALA. For meta path ASA, in Figure 10, we compare the article rankings of two recommendation lists made for the same user session using the (a) $\text{IB}_{\text{Current}}$ approach, (b) IB_{All} approach. Even though $N = 5$, $\text{IB}_{\text{Current}}$ approach is only able to recommend 3 articles, because they are the only ones

that were read together with A905. IB_{All} approach, in contrast, generates a list of articles that co-occurred in the same session with *at least one* article from current user session, thus providing recommendations that reflect the overall user interest. It is obvious, that the second approach will be more robust in terms of sparse data. This is a very intuitive and user-friendly way of explaining the relation between two articles.



(a) $IB_{Current}$ approach (b) IB_{All} approach

Figure 10: ASA-based Recommendations

In real systems, although, plain recommendations would hardly provide significant value. Instead, we can combine all four meta paths and rank them by candidate item relevance score, resulting in hybrid 4-dimensional recommendation explanations. Figure 11 shows the hybrid explanations for IB_{All} approach.



Figure 11: Hybrid 4-D meta path Explanation for IB Recommendation strategy with IB_{All} approach.

6.4 Session-based Explained Recommendations

Differently from IB recommendation strategy, building session-based (SB) explained recommendations consists of two steps:

1. Finding similar sessions;
2. Recommending the most "relevant" articles of similar sessions.

We can use three meta paths: SAS, SCS and SLS. SAS(A)-based explanation corresponds to the logical path: S $\xrightarrow{\text{contains}}$ A $\xrightarrow{\text{is read within}}$ S $\xrightarrow{\text{contains}}$ A. By combining meta paths for SB recommenations we result in hybrid 3-dimensional explanations, shown in Figure 12.

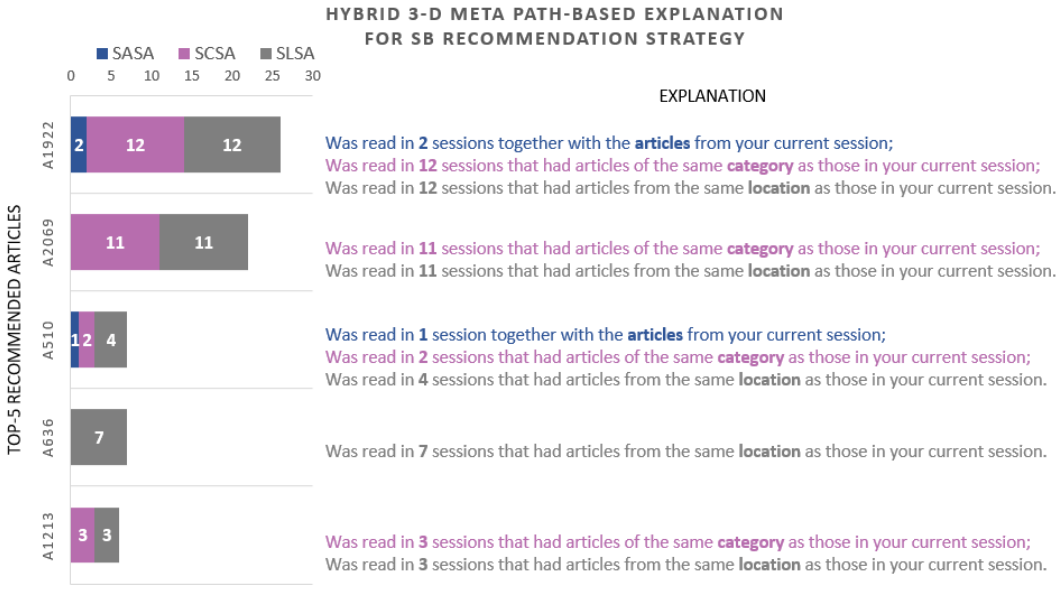


Figure 12: Hybrid 3-D meta path Explanation for SB Recommendation strategy.

6.5 User-based Explained Recommendations

User-based recommendations and explanations are very similar to session-based recommendations, with the only difference, that similar users are searched on a longer time period to identify users, that "always" coincide with their preferences. UAU(A)-based recommendation logic lays in the path: U $\xrightarrow{\text{has read}}$ A $\xrightarrow{\text{was read by}}$ U $\xrightarrow{\text{has } \textit{recently} \text{ read}}$ A

So, our training set is used for finding similar users, and we define a prediction set, for recommending news articles that are recent enough. See an example of recommendation list along with the explanation for each article.

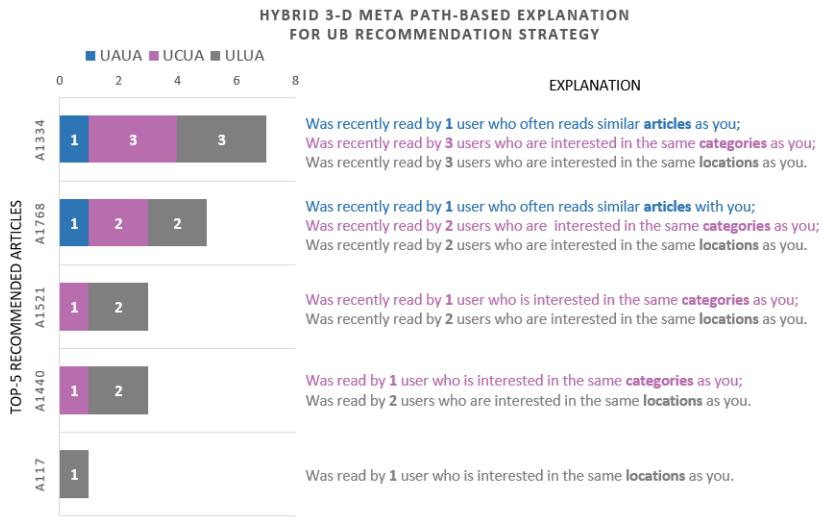


Figure 13: Hybrid 3-D meta path Explanation for UB Recommendation strategy.

6.6 Explainability coverage

We define the *explainability coverage* of a recommended item, which measures how explainable is the item recommendation to the user session and to the user. We denote the explainability of item i for a user u and his session s as $E(u,s,i)$.

DEFINITION 11. ***Explainability coverage.*** *Explainability coverage of a recommended item can be calculated as:*

$$E(u,s,i) = \sqrt{CIR_{SB}(s,i) \cdot CIR_{UB}(u,i)}, \tag{15}$$

where $CIR_{SB}(s,i)$ is a $CIR(i)$ (from Equation 14) calculated for item i for the user session s using three SB meta paths, and $CIR_{UB}(u,i)$ is a $CIR(i)$ calculated for item i for the user u using three UB meta paths.

Note, that there is no equality between possibility to explain the recommendation in natural language and the item explainability coverage score. Explainability coverage shows how strong the explanation can be if expressed in natural language, for example, both following sentences are well explained, but the second has a higher explainability score: 1) "Article A192 belongs to the same category as your currently viewed article"; 2) "Article A425 appears in 7 sessions together with items from your current session, belongs to the same category with 3 of them, and belongs to the same location with 2 of them". The explanation itself may, or may not reflect the underlying algorithm used by the system.

To the best of our knowledge there was no explainability coverage score proposed so far for recommender systems operating with implicit feedback (the first explainability for constrained matrix factorization in user-movie domain was proposed in August 2017 [2]).

7 Experimental evaluation

In this chapter we firstly present the evaluation metrics to be used in recommendations quality assessment, and describe an evaluation protocol. Next we analyze characteristics of datasets to be used in experiments. We define a popularity-based baseline, perform sensitivity analyses of all other methods defined in analytical part of this thesis, and fairly compare the methods' performances.

7.1 Evaluation metrics

This section is dedicated to the metrics used to evaluate recommendation algorithms. Historically the main attention when tuning algorithms was paid to the accuracy of the produced models, i.e. the best algorithm was the one, that most accurately guesses user interests (ratings, clicks). The two accuracy measures defined in this chapter are precision and nDCG. Nowadays, a lot of other metrics are introduced, that go far beyond accuracy. Some of those metrics even contradict with accuracy, for example, diversity of recommendation list, novelty of a recommended item to the user, recommendation explainability, and others.

7.1.1 Accuracy measures

7.1.1.1 Precision

Precision is a well known evaluation metric for binary classification algorithms, being widely applied for measuring performance of recommendation algorithms [17], suitable for both hold-out and prequential evaluation strategies. In classification and hold-out recommendation evaluation tasks, precision always goes together with *recall* measure, while for the prequential evaluation strategy these two measures coincide.

Precision usually goes with specifying a parameter k, which in prequential recommendation context stands for the length of a recommendation list produced for each test entry (i.e., for each next-click). Then, for each test entry precision is a binary value, being 1, if an entry is present in the recommendation list (*hit*), and 0, if not (*miss*).

Overall precision of the model is a proportion of number of hits (correctly predicted items within top-k lists) to total number of predictions made.

$$\text{Precision} = \frac{|\text{hits}|}{|\text{total recommendations}|} \tag{16}$$

In the field of recommender systems precision is also commonly known as *hit rate*.

7.1.1.2 Normalized discounted cumulative gain (nDCG)

nDCG is a standard ranking measure that measures the quality of produced ranked recommendation lists. It can be understood as a smoother version of precision, that takes into account the position of a correct item in the list (it is particularly important in RS as lower positioned items may be overlooked by users).

The Equation 17 is proposed to be used for prequential evaluation of RS in 2017 by Song et al. [24] as a simplified version of a standard nDCG measure.

$$nDCG = \frac{1}{log_2(1 + rank_{pos})}, \tag{17}$$

where $rank_{pos}$ denotes the rank of a positive item, i.e. the position of correctly predicted item. The lower is the relevant item in the list, the lower is the nDCG score.

7.1.2 Beyond accuracy

Accuracy of built recommendations is not the only factor that describes the performance of recommendation algorithms. What is more, aiming at solely the best accuracy of offline experiments barely leads to good recommendations and great user satisfaction when applying the model online (one of the reasons is already discussed in section 3.6 recommendation bias). There are much more measures recently proposed for measuring quality of recommendations. Some of them are listed below:

- **Diversity** - a measure of how various and diverse are the items within the recommendations list (intra-list diversity) or within different recommendation lists (inter-list diversity);
- **Novelty** - ability of a RS to recommend items that the user was not yet aware of;
- **Coverage** - percentage of the items known to the RS for which the RS can generate predictions;
- **Serendipity** - a measure of the extent to which the recommended items are both attractive and surprising to the users;
- **Confidence** - an RS ability to evaluate the likely quality of its predictions;
- **User satisfaction metrics** - an average user satisfaction level acquired with surveying the users or measuring retention and use statistics;
- **Explainability** - a measure of how explainable the recommended item is for the user, that aims at increasing transparency of a system and user trust.

Each of the above defined RS quality evaluation metrics has a number of variations, and more still continue appearing. Below we discuss one of the diversity measures, in particular - intra-list diversity (ILD), and a general concept of measuring explainability.

7.1.2.1 Intra-list diversity (ILD)

The idea of measuring diversity started from information retrieval studies, while it was measured how many senses (or subtopics) of a user query (or a topic) are covered. The earliest proposed information retrieval diversity metric was *subtopic recall* (s-recall), that is simply a ratio of query subtopics covered in the search results [3].

In the area of recommender systems, there is no direct user input (query) about what he is searching for, thus, in a recommendation list, it is more likely that the user will find an interesting item if there is a certain degree of diversity among the included items. There is often no value in having perfect recommendations for a restricted type of product, unless the user has expressed a narrow set of preferences. There are many situations, especially in the early stage of a recommendation process, in which the users want to explore new and diverse directions. In such cases, they are using the recommender as a knowledge discovery tool [21].

One of the most frequently considered diversity metrics, first proposed to use in the context of recommendation by Smyth and McClave [23], is the so-called *average intra-list distance* or just *intra-list diversity*, ILD [3]. ILD of a set of recommended items is defined as the average pairwise distance of the items in the set:

$$ILD = \frac{1}{|R|(|R|-1)} \sum_{i \in R} \sum_{j \in R} d(i,j), \tag{18}$$

where R - a recommendation list, and $d(i,j)$ - a distance measure between items i and j.

Distance measure $d(i,j)$ is a configurable element of the metric and can be defined as a complement of any well-known similarity measure between two items. In our experiments we use a simplified version of content-based item distance measure: $d(i,j) = 1$, if items i,j belong to different categories, and $d(i,j) = 0$, if to the same. This way we want to measure how diverse is the recommendation list in terms of news categories.

Summarizing, it can be said that there is a sense of contradiction between recommendation accuracy and its diversity, i.e. often the high accuracy comes with low diversity and other way around. The best trade-off between these (and more other) measures should be found in order to achieve the best quality of recommendations and user satisfaction.

7.2 Prequential evaluation protocol

The core of prequential evaluation protocol used in our experiments is inspired mainly by [26] and [10]. In [26] (2014) Vinagre et al. first proposed an RS evaluation protocol suitable for streaming data environments, but also applicable in stationary settings. It exclusively evaluates how well the model predicts the next observation, ignoring all subsequent ones, allows capturing several evaluation metrics simultaneously, and provides the possibility to monitor the evolution of algorithms' performance over time. The prequential method essentially consists of *test-then-learn* procedure that runs for each new data point of a time-ordered dataset: after a prediction is made - it is tested on a newly observed data point, and then the data point is used to update the model.

In [10] (2015) Jannach et al. explored the effectiveness of using and combining long-term and short-term adaptation strategies, and for that purpose proposed a protocol designed to evaluate both prediction accuracy and the capability of algorithms to adapt to short-term shopping goals. They used a time-related criterion to split the available data into training and test sets. For example, train the model on all but last two shopping sessions of each user, and test it on the left ones, predicting user's actions *for each session*, having in mind that his shopping goals can be different for different sessions.

As we are in the news domain, which is an extremely dynamic environment and old historical data makes no use (articles become outdated), we additionally use a sliding window approach, so that our training data always stays relevant. We distinguish between short-term, medium-term and long-term training sets, which we use for different purposes. The item-item relationships are only trained on the short-term time period. Learning the category-category and user-category preferences is performed on medium- and long-term time periods.

The schematic idea of our proposed prequential evaluation protocol is presented in Figure 14.

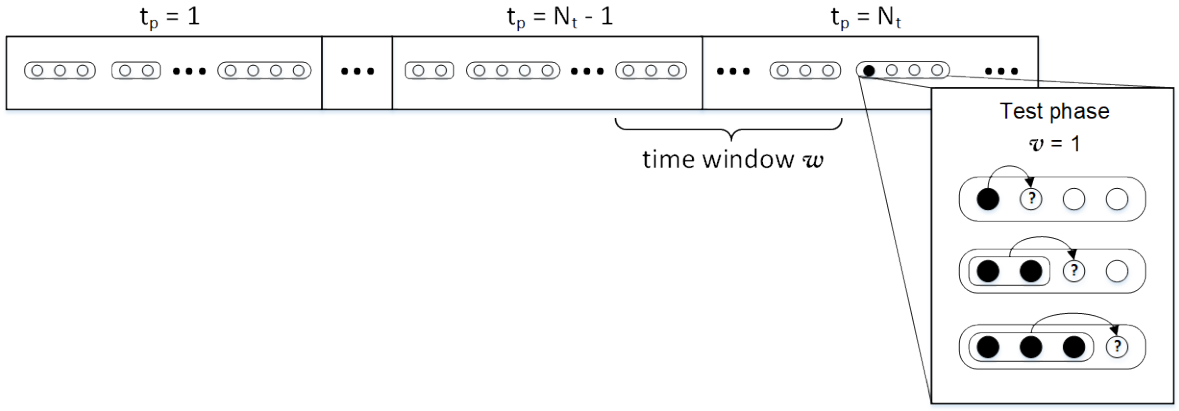


Figure 14: Prequential evaluation protocol schema

Below we list some general settings and parameters used for our proposed evaluation protocol:

1. The time frame of the available data is split into N_t equal time periods t_p, and the data is then split in the way that in each period there are only sessions that were made during that time period. We use the splitting to later aggregate the evaluation results for each time period;

2. We use parameter w to specify the window size on which the model is trained. The model training is performed for each user session, so that the most relevant information is captured every time (for example, Ludmann's news recommender system [16], denoted as Ody4, won the CLEF NewsREEL 2017 contest by just recommending the most clicked news articles of the last 12-hours time period);

3. Parameter ${}_{TOP}N$ denotes the length of the recommendation list, built for each item in a test session;

4. In the test phase we also use a parameter v to specify how many views of a currently evaluated session are revealed before recommendation is made. The parameter helps to assess how many clicks it takes for an adaptive recommendation strategy to guess the user's short-term goal. Then, after the first prediction is tested, the data point is added to the model, and the process repeats until there are data points in the session to be tested;

5. Each prediction is evaluated using defined evaluation metrics. The session is then evaluated as an average measure of made prediction for that session. The time period then gets the average evaluation of all sessions within it.

7.3 Dataset description

7.3.1 General statistics

The two datasets, used in the experiments are provided by an Italian news provider, based in South Tyrol and holding two news channels: "Video33" for Italian speaking population, and "SDF" for German speaking population. Datasets accommodate users' interactions (sessions) with news articles during one year period (from 1 April 2016 to 30 March 2017). General statistics of the datasets is summarized in Table 14, where the cleaning procedure lies in removing the sessions that contain only one article, as no predictions can be tested on such sessions, and no article co-occurence patterns can be identified. In the experiments both "before cleaning" and "after cleaning" datasets are used and compared.

	Video33		SDF	
	Before cleaning	After cleaning	Before cleaning	After cleaning
Users	14 047	918	3 626	600
Articles	2 081	1 097	468	169
Sessions	14 367	1 126	4 078	704
Avg # articles per session	1.14	2.46	1.12	2.54
Avg # articles per user	1.25	2.95	1.36	2.7
Avg # sessions per user	1.12	1.23	1.2	1.15
Avg # sessions per article	6.41	2.53	9.94	6.68

Table 14: General statistics of datasets

Figure 15 shows distributions of number of articles per session for the Video33 dataset before and after cleaning. We can see, that in the original data vast majority of sessions are very short, mostly containing one article only ("long tail" phenomenon), with the average of 1.14 articles per session. After the cleaning, the dataset is reduced drastically (more than 90% of sessions are removed), but the "long tail" is still valid. Although, the remaining sessions become longer, holding 2.46 articles on average, which is assumed to provide more meaningful results when building models.

Another observation that can be noticed from the Table 14 is that most of the users are unique in the system ("one-time"), having on average 1.12 and 1.2 sessions for two datasets, which confirms the challenge stated in the section 3.6 about lack of personal information of the uses in a news domain, as users tend to read news without authenticating themselves.

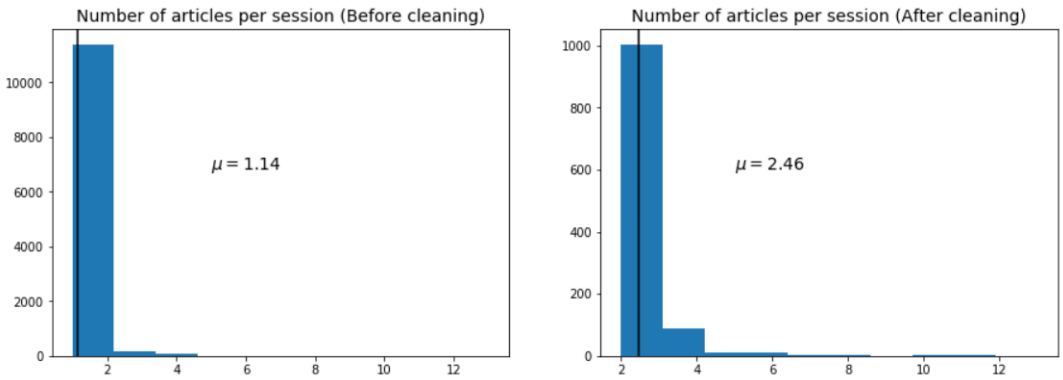


Figure 15: Distribution of number of articles per session for Video33 dataset

7.3.2 News categories

With the help of text mining techniques (in particular, by using a Latent Dirichlet Allocation (LDA) model), 5 main article topic categories are identified ("Education", "Politics", "Sports", "Local news", "Justice"), and each article is assigned to the category with a certain probability. In our experiments we restrict the analysis by assigning each article to the one most probable category. Figure 16 shows the news category preferences of users of different news channels (from different language speaking populations). One can notice, that German speaking population (SDF dataset) show an extreme interest in Justice topic category, while Italian speaking population is more interested in Politics and Local news.

Let us return to the already mentioned in section 3.5 work of Liu et al. [13], where they studied the problem of personalization of news recommendations. They discovered that individual users' interests over news categories not only change over time, but also follow those of the general public. The important observation extracted from Figure 16 is that different public communities inside one geographic region can have very different interests, and this should also be considered when building and explaining recommendations.

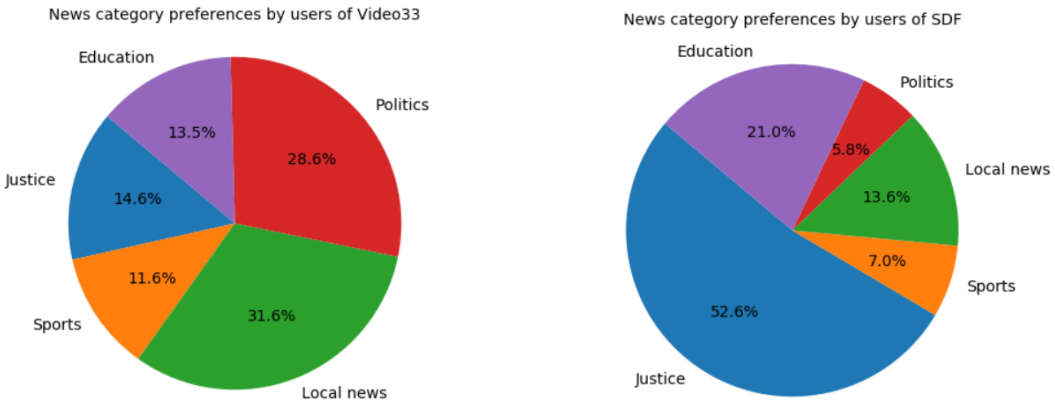


Figure 16: News category preferences by users of two news channels

The Figure 17 complies with the study of Liu et al., showing that within different periods different categories prevail over others following big events trend (for example, in June people are mostly interested in Education topic, as it is a graduation period for many of them, etc.).

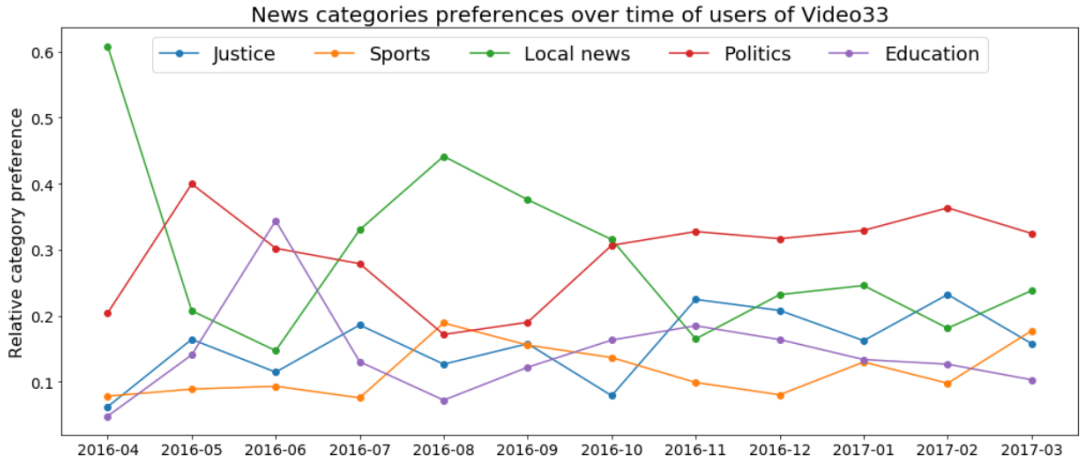


Figure 17: News category preferences dynamics at Video33 dataset

7.3.3 News locations

Journalists, producing articles for the news portal, assign them to one of the 8 locations of South Tyrol (Burgraviato/Burggrafenamt, Bolzano/Bozen, Val-d-Isarco/Eisacktal, Val-Pusteria/Pustertal, Salto-Sciliar/Salten-Schlern, Oltradige-Bassa-Atesina/Uberetsch-Unterland, Val-Venosta/Vinschgau, Alta-Val-d-Isarco/Wipptal). All location have two names (Italian and German), as the South Tyrol is a region of Italy where 70% of population are native German speakers.

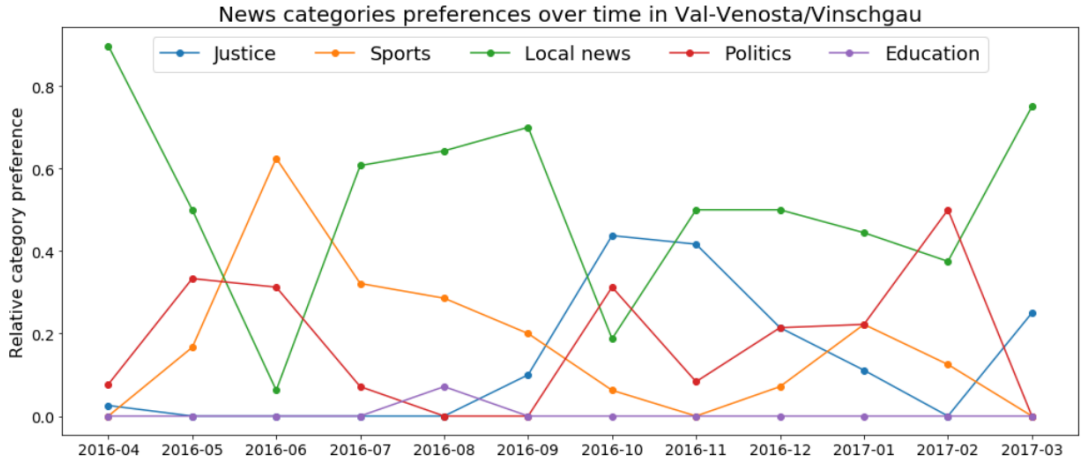


Figure 18: News category preferences dynamics at Video33 dataset in Val-Venosta/Vinschgau geographic location

Drilling-down into different geographic locations of the region we again confirm the fact, that different communities have different interests, and the category preferences of a single geographic location do not have to comply with the general preferences of the whole region. For example, consider the news category preferences of inhabitants of Val-Venosta/Vinschgau[2] presented in Figure 18. In June 2016 there was a mountain-bike marathon held in Val-Venosta, therefore we see a jump of preference in Sports category at that month.

Figure 19 distinctly shows that preferences towards different categories in different regions noticeably change over time, and rapidly enough. The first column corresponds to Bolzano/Bozen city, which is the biggest city of South Tyrol and thus shows some interest towards each category, but has different predominant categories in three subsequent months. The second column corresponds to Val-Venosta/Vinschgau, which is a small valley, and thus not every news category gets their attention, but, nevertheless, the distribution of interest over categories in three subsequent months is visibly different.

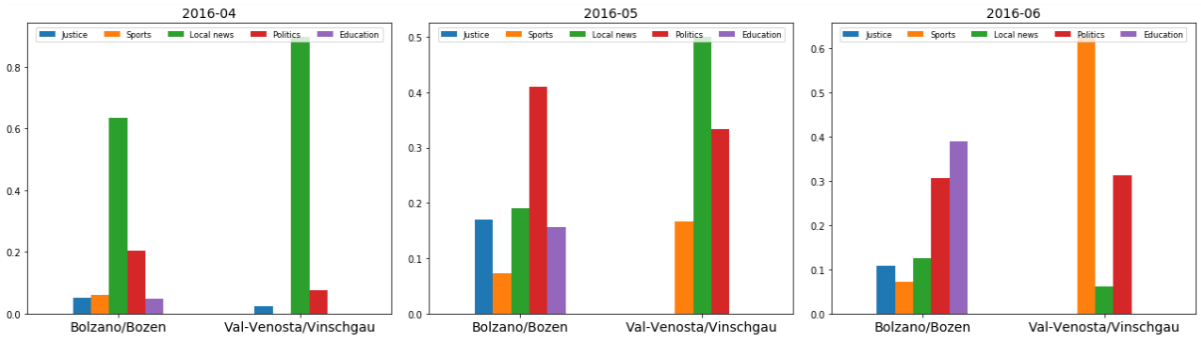


Figure 19: Category preferences in Bolzano/Bozen and Val-Venosta/Vinschgau in the period of April-June 2016

After this superficial analysis of disposable data, we can conclude, that news article category and location are its valuable additional features, that are supposed to play an important role when building next-item recommendations. We must also keep in mind, that news category preferences are changing very rapidly as news domain is a very dynamic environment. It may be of little value to recommend a news article that was popular 1 week ago. Instead, it might be reasonable to recommend an article from the category that is popular during the last week in the location that the user is currently interested in.

[2]Note, that here we assume, that inhabitants of a specific location mainly read news that are assigned to their location, and less often are interested in the news assigned to other locations.

7.4 Popularity-based baseline

In many researches it is identified that in the domain of news recommendation, the popularity-based algorithms (further - POP) show very robust results in terms of prediction accuracy, and thus are often used as a baseline to beat. In this section we run a popularity-based algorithm on original (uncleaned) data to identify which size of a short time period provides best predictions. See the evaluation results in Table 15, acquired after building 1385 requested recommendations in total. Here and further on in experiments we use the parameter $_{TOP}N = 5$ - we recommend top-5 most relevant articles for each recommendation request.

t_S	Precision	nDCG	ILD
0.5 day	0.302	0.216	0.730
1 day	**0.366**	0.256	0.743
2 days	0.325	0.219	0.729
3 days	0.265	0.171	0.723
7 days	0.145	0.095	0.708
30 days	0.048	0.031	0.708

Table 15: Popularity-based algorithm with different t_S

The best accuracy is achieved when the algorithm runs on the data from 24 recent hours. The precision of 0.366 means, that in 36.6% of cases the recommendation list contains the article that was actually read next by the user.

The recommendation list diversity in terms of topic categories of popularity-based methods is very high. While the length of our recommendation list is 5, we can identify that there are on average approximately 3.6 different categories recommended.

In Table 16 the average numbers of distinct articles and sessions per training set are shown for better understanding of the background behind the made recommendations. For example, if we want to recommend the most popular articles of the last 12 hours, we choose 5 "most popular" articles from on average 11. As long as they appear within on average 17 user sessions, it means that just few articles appeared two or three times, and others appeared only once (we have identified in Section 7.3.1 that average number of articles in a session is 1.12). Therefore, the top-5 recommendations often contain some randomly selected items, and thus the precision results slightly fluctuate with each program run.

Moreover, in a dynamic news domain articles appear very frequently, and often the articles from the test set do not appear in the train set, and thus simply cannot be recommended.

t_S	Avg # articles	Avg # sessions
0.5 day	11	17
1 day	19	33
2 days	33	62
3 days	45	90
7 days	90	204
30 days	293	869

Table 16: Average number of articles and sessions per different size of training set

Even though recommendations provided by POP are known to be accurate, they lack personalization and explainability. Thus, in next chapters we investigate, evaluate and compare other methods, that provide more relevant recommendations to users.

7.5 Sensitivity Analysis of RWR

In this section, we evaluate the performance of RWR in terms of precision, nDCG, ILD metrics and number of recommendations made, when we:

1. change the time window size $w = [0.5, 1, 2, 3, 7]$;
2. change the sub-graph used for model learning (SA, USA, SAL, etc.);
3. change the article weighting strategy (W_{LA}, W_M, W_S, W_T) discussed in Section 5.1;
4. use combinations of short, medium and long-term interests (SM, SL, SML) discussed in Section 5.2.

7.5.1 Changing the time window size w

In this section we analyze the performance of RWR when using different short-term time window sizes w. We run RWR on the whole USACL graph, weight articles in a session using W_M approach, and the number of revealed articles from the session is $v = 1$. Please note, that we evaluate the model only in case it provides a recommendation, i.e. if RWR cannot recommend any article - we do not penalize it. Therefore, for each time window size w we measure how many recommendations were made, and what percentage of the total number of recommendation requests (1385) they cover. The performance results are shown in Table 17.

As we can see from the last two columns of Table 17, the smaller is the window size - the less recommendations can be made (i.e., the model is learned on only a small amount of the most recent news). The less recommendations are made - the less reliable the precision metric is. For example, comparing the precision of RWR on 12 last hours and

w	Precision	nDCG	ILD	# Recs	% Recs
0.5 days	0.386	0.278	0.586	806	58%
1 day	**0.400**	0.289	0.496	910	66%
2 days	0.398	**0.298**	0.444	967	70%
3 days	0.384	0.289	0.456	1014	73%
7 days	0.358	0.268	0.468	1054	76%

Table 17: RWR evaluation by using different window sizes w

on 3 last days (1st and 4th row of the Table 17) we can state that the later is more precise and robust, as long as it can provide more recommendations with the same probability of being accurate in prediction (the small difference is due to chance). The window size that provides the highest precision is 1 day, however, only 66% of recommendations are evaluated, while others are not fulfilled. The precision of RWR run on 2 recent days is very close (we could even say the same), however the number of predictions made is higher. Moreover, the nDCG of RWR run on 2 days is the highest.

In further experiments we use $w = 2$, as to our belief it provides the best accuracy/recommendation coverage ratio, and semantically suits the news recommendation problem - not too recent, and not too old articles are recommended.

7.5.2 Changing the network structure G

In this section, we evaluate the performance of RWR when we run it on different sub-graphs that can be derived from graph USACL. The configuration parameters of RWR: time window $w = 2$, articles weighting approach W_M, number of revealed articles from the session $v = 1$. Please note that similarly to POP, the acquired results are not always stable and slightly fluctuate accidentally with each program run. Nevertheless, some general observations about the impact of each network participating entity can still be made. The performance results of the RWR models based on different sub-graphs are provided in Table 18 in the descending order of precision.

As shown in Table 18, the three best results of RWR are acquired using very similar sub-graphs SAL, UAL, and USAL. The thing, that catches the eye, is that sub-graphs containing location entity (L) perform better than those containing category (C). And even more, considering the category reduces the accuracy comparing to sub-graphs with location only (SAL > SACL > SAC, UAL > UACL > UAC, USAL > USACL > USAC). This brings to the mind an idea that in the news domain people less care about reading the news from one specific category in one session (e.g. "Today I want to inform myself about all sports news in the country"), but are more interested in what's new in their specific region (e.g. "What did happen today in my city?").

Sub-graph	Precision	nDCG	ILD
SAL	**0.436**	**0.316**	0.744
UAL	**0.432**	**0.313**	0.743
USAL	**0.429**	**0.315**	0.741
SACL	0.403	0.303	0.460
UACL	0.397	0.299	0.478
USACL	0.392	0.297	0.449
SAC	0.389	0.297	0.405
USAC	0.388	0.297	401
UAC	0.385	0.295	0.424
USA	0.369	0.278	0.728
UA	0.365	0.272	0.728
SA	0.358	0.270	0.730
ACL	0.140	0.080	0.157
AL	0.139	0.081	0.725
AC	0.123	0.072	0.107

Table 18: RWR evaluation by using different network structures G

We can also notice, that the category-based diversity (ILD) of more accurate models is very high, which conflicts with the commonly accepted claim that in recommendation systems accuracy and diversity measures contradict each other.

Another observation from Table 18 is that there is almost no difference between the impact of session (S) and user (U) entities. The four top blocks consisting of three lines have the same ending (AL, ACL, AC, A), and their beginnings differ between S, U, and US. This effect is easily explainable, as long as during the dataset analysis in Section **??** we figured out, that most of the users have only 1 session, and therefore S≈U.

Finally, from the two bottom blocks of the Table 18 we verify, that collaborative filtering based approaches (USA, UA, SA) are performing much better than content (feature) based ones (ACL, AL, AC). This is also as expected, because content-based approaches in our case are very close to random, while they randomly select 5 articles from the same category or location (or both) to form a recommendation list, without any additional ranking criteria.

7.5.3 Changing the weighting strategy of articles in a session

In this section, we perform an experiment to clarify if there is a noticeable difference between the four article weighting approaches described in Section 5.1. For the training we use the all available data, whereas for the test phase we perform two additional steps:

1. *Test sessions cleaning*: we remove the articles that do not appear in the training set (no relevance vectors exist for them). This way the comparison is more accurate: for example, there are no sessions of length 3, in which only 1 article has appeared in the training and, thus, only 1 relevance vector is considered in all four approaches;

2. *Establishing parameter* $v = 2$: we start making recommendations after revealing two consequent user view actions (from already filtered sessions). The reason for this is that we want to objectively compare the influence of different article weighting approaches, while the prediction for the first article in a session should be the same for all approaches.

After performing these two steps, the average number of articles per test session is 3.4 articles, which is a good basis for comparing article weighting strategies.

We use window size $w = 2$, and run the algorithm on the whole USACL graph. Experimental evaluation results are presented in Table 19. Number of recommendations made and evaluated - 146.

Weighting strategy	Precision	nDCG	ILD
W_{LA}	**0.481**	**0.339**	0.400
W_M	0.457	0.330	0.564
W_S	**0.478**	**0.337**	0.526
W_T	0.420	0.319	0.474

Table 19: RWR evaluation by using different article weight assignment strategies W

As shown in Table 19, the two best performing strategies are W_{LA} and W_S, which are both biased towards the last article in the session. The 3rd strategy in terms of accuracy is W_M, and the least accurate is W_T. On the other hand, we see that diversity of W_{LA} is lower than others, while W_M provides the highest diversity. As long as W_{LA} is not a personalized approach and only searches for similar items to the last article, we will further use the W_S strategy to capture the whole user session, but treat the more recent items as more important ones.

We must mention here that as the dataset size is highly reduced after the cleaning step (i.e. removing sessions with 1 item only), there might be not enough data to reliably distinguish between strategies. Even if their produced item relevance scores and, consequently, their item rankings are different, the top-5 recommended items are often the same.

7.5.4 Combining short-, medium- and long-term user interests

In this section, we show the experimental results of RWR when we use combinations of short, medium and long-term user interests (SM, SL, SML) discussed in Section 5.2. We use three window sizes for short, medium and long terms, in particular 2, 7, and 30 days correspondingly. We build recommendations only for active users, i.e. for those users from the test set, that appeared in the long-term training set. 290 recommendations were made and evaluated.

Table 20 shows, that, unfortunately, no improvement is achieved when the medium and long-term object relationships are considered. Vice versa, both the precision and diversity decrease. Thus, this method is not taken into account any further. The best performance is attained by capturing simply the short-term user actions. This can be explained by the fact that news articles appear and die very quickly in a dynamic environment of news domain.

Approach	$[w_S, w_M, w_L]$	Precision	nDCG	ILD
S	[1, 0, 0]	**0.394**	0.259	0.403
SM	[0.7, 0.3, 0]	0.271	0.188	0.109
SL	[0.7, 0, 0.3]	0.278	0.191	0.099
SML	[0.7, 0.2, 0.1]	0.276	0.191	0.104

Table 20: Comparison of S, SM, SL and SML approaches

In addition, we show in Figure 20 the dynamics of four approaches performance over 11 time periods (months). As can be seen, S strategy is constantly above the other comparison strategies.

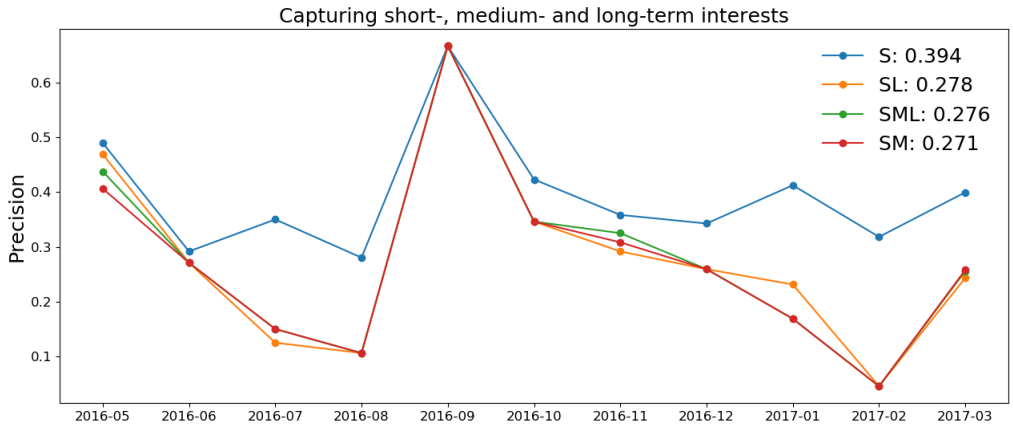


Figure 20: Comparison of combinations of short-, medium- and long-term interests

7.6 Sensitivity analysis of PathSim and PathCount

In this section, we evaluate the performance of PathSim and PathCount in terms of precision, nDCG, ILD metrics and number of recommendations made, when we:

1. change time window size $w = [0.5, 1, 2, 3, 7]$;
2. change the meta paths used for model learning (AUA, ASA, ACA, ALA);
3. change the article weighting strategy (W_{LA}, W_M, W_S, W_T) discussed in Section 5.1.

7.6.1 Changing the time window size w

In this section, we analyze the performance of PathSim and PathCount when using different time window sizes w. The other configuration parameters are kept as: meta path ASA, articles weighting strategy W_M, number of revealed articles $v = 1$. The performance results are shown in Table 21.

w	Precision		nDCG		ILD		# Recs	% Recs
	PathSim	PathCount	PathSim	PathCount	PathSim	PathCount		
0.5 days	0.365	0.367	0.250	0.260	0.714	0.713	793	57%
1 day	0.338	0.340	0.248	0.257	0.723	0.722	930	67%
2 days	**0.340**	**0.344**	0.248	0.259	0.730	0.730	970	70%
3 days	0.313	0.316	0.235	0.243	0.722	0.723	1010	73%
7 days	0.281	0.283	0.218	0.224	0.703	0.702	1060	77%

Table 21: PathSim and PathCount evaluation by using different window sizes w

From the Table 21 we can see that 2 days time window provides the best precision with 70% of recommendation requests fulfilled. Moreover, we can observe that with any time window our proposed PathCount, even while performing almost the same as PathSim, shows a very small but constant superiority over PathSim in terms of both precision and nDCG.

7.6.2 Changing the meta path

In this section, we evaluate the performance of PathSim and PathCount when running it on four possible meta paths, that start and end at the article node: AUA (a shortening of ASUSA), ASA, ALA, ACA. We use time window $w = 2$, weighting strategy W_M and number of revealed articles $v = 1$. Similarly to the RWR evaluation, we evaluate the recommendation of each model only when all the models have something to recommend. There were 848 recommendations made and evaluated in total. The evaluation results are shown in Table 22.

Meta path	Precision		nDCG		ILD	
	PathSim	PathCount	PathSim	PathCount	PathSim	PathCount
AUA	0.332	**0.333**	0.242	0.253	0.726	0.729
ASA	0.331	**0.333**	0.244	0.254	0.744	0.735
ACA	0.114	0.111	0.063	0.061	0.111	0.105
ALA	0.133	0.316	0.073	0.076	0.724	0.724

Table 22: PathSim and PathCount evaluation by using different meta paths

From Table 22 we cannot observe a noticeable difference in precision of models based on AUA or ASA meta paths. Also, we don't see the superiority of PathCount in terms of precision, but 1% superiority in terms of nDCG. What is more, we see that using content-based meta paths (ACA and ALA) we result in a 3 times lower accuracy than with collaborative filtering related meta paths (AUA and ASA). This is as expected, because, when using PathSim and PathCount measures, all the articles that belong to the same category or location get similarity score 1 (for PathSim) or equal transition probabilities (for PathCount). This way, a top-5 recommendation list consists of 5 random articles having the same feature value.

7.6.3 Changing the weighting strategy of articles in a session

In this section, we compare PathSim and PathCount methods when changing the article weighting strategy. We perform the same two additional data preparation steps as described in section 7.5.3 for RWR. The configuration parameters are: time window $w = 2$, meta path AUA, number of revealed articles $v = 2$. Same as for RWR, there were 146 recommendations made and evaluated. The experimental results are presented in Table 23.

Weighting strategy	Precision		nDCG		ILD	
	PathSim	PathCount	PathSim	PathCount	PathSim	PathCount
W_{LA}	0.316	0.317	0.230	0.237	0.702	0.713
W_M	0.354	0.370	0.256	**0.297**	0.674	0.674
W_S	0.354	**0.380**	0.259	0.295	0.677	0.674
W_T	0.353	0.366	0.265	0.293	0.674	0.674

Table 23: PathSim and PathCount algorithms by using different article weight assignment strategies W

From Table 23 we can notice that, in contrast to RWR's experimental results (Table 19), W_{LA} performs noticeably worse than other weighting strategies which consider all items inside a user session (while for RWR the W_{LA} was the best). The reason for

this is that the RWR algorithm, when learning, naturally captures information from item neighbors when building a relevance matrix, thus, even when we only consider one item, more items get some relevance because of one or another reason. PathSim and PathCount are more restricted to one particular meta path and therefore a lot of items get zero relevances. As a result, PathSim or PathCount-based methods usually can only build shorter recommendation lists than the stated $_{TOP}N$.

Another observation is that PathCount consistently performs better than PathSim in terms of accuracy, which confirms our claim that it fits better the news application domain.

7.7 Sensitivity analysis of Session-kNN

In this section, we evaluate the performance of Session-kNN when using RWR and PathCount similarity measures. We evaluate the algorithm with different parameters of similarity search methods and with different numbers of k nearest sessions.

7.7.1 Changing the similarity search model

In this section we evaluate Session-kNN when changing similarity search model. The experimental results are shown in Table 24 in descending order of precision. There were 646 recommendations made.

Similarity	Precision	nDCG	ILD
RWR_{SAL}	**0.490**	0.261	0.726
RWR_{USAL}	0.484	0.255	0.722
RWR_{USA}	0.474	0.251	0.720
RWR_{SAC}	0.463	0.249	0.553
RWR_{SA}	0.459	0.247	0.723
RWR_{USACL}	0.458	0.246	0.561
RWR_{USAC}	0.454	0.243	0.545
$PathCount_{SAS}$	0.425	**0.350**	0.525
$PathCount_{SALAS}$	0.112	0.078	0.777
$PathCount_{SACAS}$	0.085	0.050	0.157

Table 24: Session-kNN evaluation by using different similarity search models

From the Table 24 we can observe, that PathCount is less accurate than RWR in terms of precision. However, PathCount on meta path SAS shows the best nDCG measure, i.e. when the right article is in the recommendation list, it usually appears higher in the list that in the RWR case.

7.7.2 Changing the number of nearest sessions k

In this section we evaluate the performance of Session-kNN when changing the number of nearest sessions taken into account. We test the k parameter on values [1, 2, 3, 5, 7, 9, 15, 30], while there is on average 61 session per 2 days training set. 646 recommendations were made in total. The experimental results are provided in Table 25.

k	Precision		nDCG		ILD	
	RWR_{SAL}	$PathCount_{SAS}$	RWR_{SAL}	$PathCount_{SAS}$	RWR_{SAL}	$PathCount_{SAS}$
1	0.498	0.254	0.262	0.240	0.727	0.203
2	0.495	0.357	0.262	0.307	0.721	0.510
3	0.494	0.408	0.262	0.336	0.722	0.540
5	0.490	0.425	0.261	0.350	0.726	0.525
7	0.477	0.428	0.256	0.350	0.719	0.549
9	0.492	0.445	0.264	0.368	0.720	0.565
15	0.480	0.436	0.259	0.360	0.715	0.550
30	0.493	0.443	0.266	0.364	0.721	0.542

Table 25: Session-kNN precision evaluation by using different k

From the table we can observe, that when increasing the k from 1 till 7, the RWR precision falls down, while the PathCount precision goes up (but still doesn't rich the RWR's precision). As was already mentioned, RWR naturally captures in information from nearest neighbors when inferring similarities, thus just taking the information from the item relevance vector of the one nearest session appears to be enough. While PathCount only captures information of directly connected items, thus to consider more items, more neighboring sessions should be used. Summarizing, we can say that Session-kNN based on RWR is not only more accurate, but also more effective in terms of computations.

7.8 Methods comparison

In this section, we present the comparison of methods. We take two RWR models: 1) based on SAL sub-graph as the most accurate, and 2) based on USACL graph as capturing the most information. We also perform Session-kNN based on these two RWR models. As well as Session-kNN's on PathSim and PathCount based on meta path SAS. We also compare plain PathSim and PathCount on meta paths AUA and ASA. Here we also for the first time show the experimental performance of SimRank. Popularity baseline is also included. Moreover, we run these models on two datasets to make more reliable conclusions.

There were 642 recommendations made for Video33 dataset, and 257 - for SDF dataset. See Table 26 for experimental results.

Model	Video33			SDF		
	Precision	nDCG	ILD	Precision	nDCG	ILD
RWR_{SAL}	**0.545**	0.399	0.738	**0.683**	0.563	0.547
RWR_{USACL}	**0.499**	0.381	0.558	**0.691**	0.550	0.418
Session-kNN(RWR_{SAL})	**0.497**	0.267	0.730	0.655	0.344	0.354
$PathCount_{AUA}$	0.484	0.357	0.729	0.674	0.522	0.629
$PathSim_{AUA}$	0.477	0.370	0.733	0.644	0.487	0.629
$PathCount_{ASA}$	0.461	0.362	0.734	0.671	0.534	0.631
$PathSim_{ASA}$	0.458	0.341	0.734	0.641	0.500	0.632
Session-kNN(RWR_{USACL})	0.454	0.244	0.552	**0.683**	0.536	0.547
Session-kNN($PathCount_{SAS}$)	0.443	0.362	0.549	0.585	0.495	0.117
Session-kNN($PathSim_{SAS}$)	0.443	0.361	0.552	0.585	0.497	0.113
POP	0.437	0.303	0.735	0.552	0.446	0.676
$SimRank_{SAL}$	0.378	0.291	0.733	0.614	0.442	0.527
# Recs	642			257		

Table 26: Models comparison

From Table 26 we can observe that on different datasets different models perform differently. While for Video33 Session-kNN based on the whole USACL graph performs quite bad, for the SDF it is the second best. However, the general observation is that we are able to beat the popularity baseline with RWR, PathSim and PathCount methods. The only $SimRank_{SAL}$ performs worse (we have also run SimRank on other sub-graphs, but in the thesis volume constraint we present here only the best performing one).

In Figure 21 we present a visual comparison of models over 12 months. We can notice, that even though the precision fluctuates in each period (depends on the number of recommendations made in each period), but the relation between different models' precision stays similar - RWR_{SAL} is always above the others except one period 2016-09, where the POP shows better, as long as POP shows the most unpredictable performance over the year.

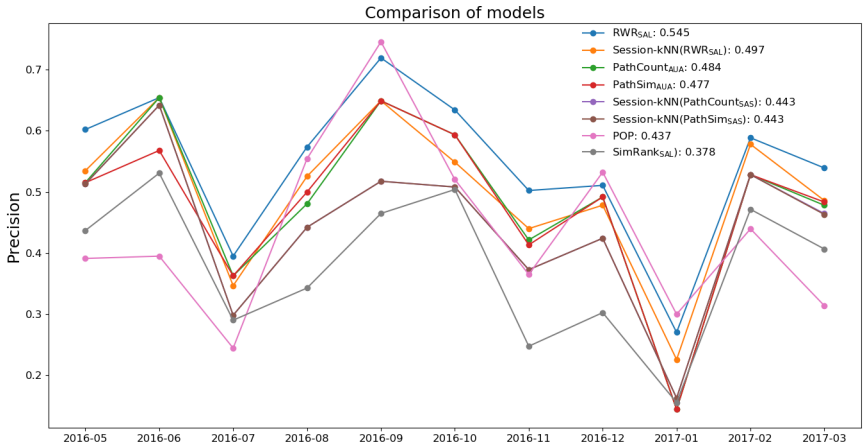


Figure 21: Models precision comparison on Video33 dataset

7.9 Experiments on explainable recommendations

In this section, we provide experimental results on the explainable recommendations strategies, described in Section 6. For each strategy we compare the results achieved by each plain meta path and by their combination. We use the 2 days short-term time window for AB and SB approaches, and for UB approach we learn the model on 1 month of data (find similar users over the month) and recommend the recently (within 2 days) read articles from similar users. Moreover, we provide for comparison also the results of $\text{RWR}_{\text{USACL}}$, Session-kNN($\text{RWR}_{\text{USACL}}$) and POP. We compare the models based on precision and explainability coverage (Equation 15). Experimental results are presented in Table 27. Total 645 recommendation were evaluated.

Metric	Model	Item-based Rec. ($\text{IB}_{\text{Current}}$)				Item-based Rec. (IB_{All})				Session-based Rec. (SB)			User-based Rec. (UB)		
		AUA	ASA	ACA	ALA	AUA	ASA	ACA	ALA	SASA	SCSA	SLSA	UAUA	UCUA	ULUA
Precision	Single meta path	0.433	0.423	0.142	0.143	0.440	0.431	0.128	0.150	0.431	0.357	0.418	0.448	0.287	0.350
	Multiple meta paths	0.476				0.477				**0.522**			**0.500**		
	$\text{RWR}_{\text{USACL}}$	0.499													
	Session-kNN($\text{RWR}_{\text{USACL}}$)	0.460													
	POP	0.448													
Explain. Cov.	Single meta path	0.363	0.374	0.129	0.108	0.366	0.376	0.130	0.110	0.376	0.242	0.262	0.351	0.216	0.241
	Multiple meta paths	0.242				0.249				**0.310**			**0.293**		
	$\text{RWR}_{\text{USACL}}$	0.274													
	Session-kNN($\text{RWR}_{\text{USACL}}$)	0.243													
	POP	0.261													

Table 27: Explainable recommendations precision and explainability coverage vs. different explanation styles and recommendation strategies

We can see from the Table 27 that IB_{ALL} approach shows slightly better than $\text{IB}_{\text{Current}}$, i.e. capturing the whole user session is normally better than the last article only. We also see that IB_{All}(ASA) shows the same as SB(SASA), which is as expected.

We can also observe that for each recommendation strategy, recommendations based on combined meta paths are more accurate than based on any single meta path. The best accuracy provides the Session-based strategy, followed by User-based. Both SB and UB show better accuracy and explainability than RWR and Session-kNN.

Even though, RWR, Session-kNN and POP models show high enough explainability coverage score, additional steps must be performed to actually transform the explanations into natural language, while our proposed methods are made to easily explain recommendations, but keeping also high accuracy level.

8 Results and Conclusions

The **results** of this master's thesis are summarized as follows:

1. Existing graph-based recommendation techniques, in particular SimRank, Random Walk with Restart, PathSim, Session-kNN, are studied in detail, evaluated and compared experimentally on different configuration settings. A variation of PathSim denoted as PathCount is proposed as better applicable for news application domain;

2. The means of generating explainable recommendations are explored and three recommendation strategies - item-based, session-based, and user-based - are proposed to generate both accurate and explainable recommendations using graph meta paths;

3. The popularity baseline was beaten by most of the considered methods.

The **conclusions** of this master's thesis are summarized as follows:

1. The proposed similarity metric PathCount experimentally proven to be slightly more accurate than PathSim, both when run directly to build recommendations, and when integrated into Session-kNN method for sessions similarity search. However, the RWR in all cases outperforms both PathSim and PathCount;

2. The location is a very important participating entity in the news domain and mostly conditions the high accuracy level - it is important for news readers to read news from their interesting location;

3. When using RWR, the best accuracy is achieved by considering only the last item in the session. When using PathSim or PathCount, considering only the last article leads to the lowest accuracy. The best accuracy is achieved with either equally treating all session articles, or giving slightly more weight to the articles that are closer to the end of the session;

4. Combining medium- and long-term object relationships with short-term user interests reduces the accuracy comparing to short-term based algorithm alone;

5. In Session-kNN, bigger k reduces the accuracy of RWR, but increases the accuracy of PathSim and PathCount;

6. Session-based and user-based explainable recommendation strategies show the best results in terms of both accuracy and explainability coverage, being initially developed to be able to produce natural language recommendation explanations.

References

[1] The web conference. lyon, france. 2018.

[2] B. Abdollahi and O. Nasraoui. Using explainability for constrained matrix factorization. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17. ACM, 2017.

[3] P. Castells, N.J. Hurley, and S. Vargas. Novelty and diversity in recommender systems. 2015.

[4] E.V. Epure, B. Kille, and J.E. Ingvaldsen. Recommending personalized news in short user sessions. *ACM*, 2017.

[5] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". 2016.

[6] J.A. Gulla, B. Yu, Ö. Özgöbek, and N. Shabib. Third international workshop on news recommendation and analytics (inra 2015). In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 345–346, 2015.

[7] T.H. Haveliwala. Topic sensitive page rank. *ACM*, 2002.

[8] B. Hidasi, L. Baltrunas, A. Karatzoglou, and D. Tikk. Session-based recommendations with recurrent neural networks. 2016.

[9] D. Jannach and M. Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 306–310, New York, NY, USA, 2017. ACM.

[10] D. Jannah, L. Lerche, and M. Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. *ACM*, 2015.

[11] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2002)*, pages 538–543, Edmonton, Canada, 2002.

[12] J. Leskovec, A. Rajaraman, and J. Ullman. *Mining of Massive Datasets*. 2014.

[13] J. Liu, P. Dolan, and E.R. Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces, IUI'10*, pages 31–40, 2010.

[14] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang. Stamp: Short-term attention/memory priority model for session-based recommendation. 2016.

[15] M. Ludewig and D. Jannah. Evaluation of session-based recommendation algorithms. 2018.

[16] C Ludmann. Recommending news articles in the clef news recommendation evaluation lab with the data stream management system odysseus. In *Working Notes of the 8th International Conference of the CLEF Initiative, Dublin, Ireland. CEUR Workshop Proceedings*, 2017.

[17] M. Malaeb. Recall and precision at k for recommender systems, 2017.

[18] L. Page and S. Brin. The pagerank citation ranking: Bringing order to the web. 1998.

[19] A. Papadimitrou, P. Symeonidis, and Y. Manolopoulos. A generalized explanations styles taxonomy for the traditional and the social recommender systems. *Data Min Knowl Disc*, 2012.

[20] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM*, 2018.

[21] F. Ricci, L. Rokach, and B. Shapira. Recommender systems: Introduction and challenges. 2015.

[22] B. Sarwar, G. Karypis, and J. Konstan, J. an Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, pages 285–295, New York, NY, USA, 2001.

[23] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings of the 4th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, pages 347–361, London, UK, 2001.

[24] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang. Session-based social recommendation via dynamic graph attention networks. *ACM*, 2017.

[25] G. Sottocornola, P. Symeonidis, and M. Zanker. Session-based news recommendations. 2018.

[26] J. Vinagre, A.M. Jorge, and J. Gama. Evaluation of recommender systems in streaming environments. 2014.

[27] S. Yizhou, H. Jiawei, Y. Xifeng, Y. Philip S., and W. Tianyi. Pathsim: Meta path-based top-k similarity search in heterogenuos information networks. In *Proceedings of the VLDB Endowment (VLDB'2011)*, Seattle, Washigton, 2011.